Universidad Carlos III de Madrid

Escuela Politécnica Superior

Grado en Ingeniería Electrónica Industrial y Automática

TRABAJO FIN DE GRADO

Modelado de robot submarino mediante el simulador UWSim

Autor: Isaac Sánchez Rodríguez

Tutor: Alberto Jardón Huete

Departamento de Ingeniería de Sistemas y Automática

Leganés, septiembre 2017.

AGRADECIMIENTOS

A mi familia, por su apoyo y paciencia interminables durante estos años. Sin ellos no hubiese sacado ganas e ilusión en varios momentos difíciles para seguir adelante.

A mis amigos de la Universidad, por sufrir junto a mí esos momentos y hacer que tenga un muy buen recuerdo de mi vida universitaria.

A mis amigos “de toda la vida” por apoyarme como lo han hecho desde que éramos pequeños.

RESUMEN

Este Trabajo Fin de Grado forma parte del Proyecto STAMS, un consorcio entre la Universidad Carlos III de Madrid y varias empresas privadas para desarrollar las herramientas robóticas necesarias para la monitorización de minas inundadas. De dicho proyecto surgen varios Trabajos de Fin de Grado, siendo este uno de ellos.

En este trabajo se busca la integración del ROV BlueROV2, fabricado por Blue Robotics, en el entorno de simulación de robótica submarina UWSim, siendo capaz de controlarlo en dicho entorno y tratar la información que generen sus sensores.

Para ello es necesario implementar el modelo 3D, y las herramientas necesarias para su funcionamiento, como sensores y actuadores. Así mismo, se implementarán también los mecanismos de control de los mismos, para el análisis e investigación requeridos por el usuario final.

Este proyecto se desarrollará principalmente bajo el Sistema Operativo Linux, con el software UWSim y ROS en su versión Indigo.

ABSTRACT

This Bachellor thesis is part of the STAMS project, collaboration between Universidad Carlos III and several private companies, to develop the robotics tools required for monitoring flooded mines. This project gives rise to different Bachellor thesis, this being one of them.

In this work we seek the integration of ROV BlueROV2, manufactured by Blue Robotics, in UWSim, a simulation environment of underwater robotics, being able to control it in that environment and use the information generated by its sensors.

This requires the implementation of the 3D model, and the necessary tools for its operation, such as sensors and actuators. Likewise, the control mechanisms will also be implemented for the analysis and research required by the end user.

This project will be mainly developed under the Linux Operating System, with the software UWSim and ROS in its Indigo version.

ÍNDICE

1. Introducción

1.1. *Remote Operated Vehicles* (ROV)

Las siglas ROV hacen referencia a cualquier vehículo operado remotamente, independientemente del medio en el que lo haga. No obstante, estas siglas están ligadas a aquellos vehículos que operan en un ambiente subacuático.

Las siglas exactas para los vehículos submarinos serían ROUV (*Remote Operated Underwater Vehicle*) o UUV (*Unmanned Underwater Vehicles*), aunque por tradición, o por el hecho de que los vehículos que operan en otros medios reciben otras siglas (UAV para los operados remotamente en el aire y RCV para los que operan en medio terrestre), se habla de ROV para referirse a los ROUV/UUV.

Los vehículos operados remotamente son usados en multitud de tareas como el rescate de personas en catástrofes naturales, operaciones militares, mapeado de terrenos terrestres y submarinos o revisión de cables subacuáticos.

La consigna básica de estos vehículos es la de disponer un sistema manejable, que pueda ser controlado por el operador desde una localización externa. Dicho control puede realizarse a través de un cable que los una (el cual provee de energía y señal al vehículo) o mediante radiofrecuencia u otro tipo de señal inalámbrica (este sistema no es empleado en vehículos subacuáticos debido a la dificultad de propagación de las ondas a través del agua).

Debido a la naturaleza de este trabajo, el foco se centrará sobre los ROV acuáticos.

Estos vehículos están equipados con propulsores que permiten el movimiento (normalmente 6, dispuestos en parejas para cada movimiento en los ejes principales XYZ, además de las rotaciones RPY), cable de alimentación de energía y señal de control, y diversos sensores, entre los que se encuentran principalmente cámaras de vídeo (para la retransmisión en directo de la situación del vehículo).

Asimismo, pueden estar equipados con elementos como sónar (para realizar mapeados del entorno) o brazos robotizados (para tareas como recogida de muestras).

1.2. Motivación del presente trabajo

El proyecto STAMS se enmarca dentro de la *Research Foundation for Coal and Steel*, financiado por la UE. Entre las empresas del consorcio destacan INERIS como líder, y Grupo Hunosa y KWSA como clientes finales, dos de los mayores propietarios de minas de Europa.

La motivación de este proyecto surge del hecho de que cada vez más minas se cierran por toda Europa. La mayoría de ellas están inundadas al estar atravesadas por acuíferos, lo que puede implicar la contaminación de los mismos y posibles derrumbes por la corrosión de los materiales de sujeción.

Para ello se busca implementar soluciones para monitorizar periódicamente las minas y pozos inundados durante un período prolongado, creando los módulos necesarios que tomen medias mediante sensores de ultrasonidos (debido a las condiciones de baja visibilidad).

Se incluye también el desarrollo del software necesario para controlar todo el proceso y que analice los datos obtenidos por los módulos.

En este contexto, una de las partes necesarias, y motivación de este trabajo, es la del modelado del robot BlueROV2 en el software de simulación submarina UWSim, para posteriores investigaciones y simulaciones, debido a que el fabricante del ROV no provee de un simulador propio.

Para dicho fin se ha de crear tanto el modelo 3D del vehículo, como las herramientas software que permitan su control y adquisición de datos.

1.3. Estructura del documento

Este documento comienza con un análisis del estado del arte de la robótica aplicada a vehículos operados remotamente. En este capítulo se analizan tanto los antecedentes de los vehículos submarinos operados en remoto, categorías distinguibles, el estado actual de los mismos y las consideraciones de diseño que poseen.

A continuación se analizan las herramientas informáticas utilizadas, con una breve descripción de las características más destacables de cada una.

Se contempla en un capítulo específico el marco regulatorio que abarca al proyecto, indicando las pautas legales más importantes a tener en cuenta, y del que se extrae la idea de que aún no existe un estatus jurídico bien definido que aborde el uso de este tipo de vehículos.

El grueso del trabajo se centra en el vehículo a integrar, el trabajo desarrollado para dicho modelado (archivos creados, y explicación de los mismos), reservando un capítulo para el análisis de los resultados obtenidos y otro para las futuras mejoras que se pueden realizar para ampliar y perfeccionar el proyecto.

Posteriormente se realiza un estudio del impacto socioeconómico que supondría la implantación de este trabajo, así como el presupuesto necesario para llevarlo a cabo.

Varios anexos se sitúan al final del documento para ampliar la información sin entorpecer demasiado la lectura del documento principal, y 2 índices, uno de figuras y otro de tablas empleadas para facilitar la búsqueda de las mismas.

Las últimas páginas se reservan para detallar la bibliografía empleada para la realización del trabajo, detallando la fuente utilizada y la fecha de última visita en caso de que la fuente sea una página web, e índice de figuras y tablas, para facilitar su búsqueda en el documento.

2. Estado del arte

2.1. Antecedentes

Los primeros desarrollos de ROVs se dieron por parte de la marina de Estados Unidos, la cual desarrolló el CURV (*Cable-Controlled Underwater Recovery Vehicle)*. Las funciones principales del CURV eran las de realizar operaciones de rescate en alta mar (como el salvamento de los pilotos de un sumergible hundido a 500m de profundidad con pocos minutos de aire restante en Cork, Irlanda, en el año 1973) y recuperación de objetos del fondo marino, como la bomba nuclear recuperada en Palomares, España, en el año 1966 tras el choque de dos aviones en la zona [1].

Figura 1. CURV III desarrollado por la Marina estadounidense [2].

Los siguientes avances fueron principalmente financiados por empresas privadas (mayoritariamente petroleras) que vieron en los ROV una oportunidad para desarrollar nuevos pozos de extracción de crudo.

Los primeros modelos en ver la luz con el anterior propósito fueron el RCV-225 y el RCV-150, desarrollados por la empresa norteamericana HydroProducts.

Figura 2. ROV RCV-225 de la empresa HydroProducts[3].

Ya en la década de 1980 los ROVs se convirtieron en una pieza fundamental en las operaciones de desarrollo submarino, ya que estas excedían la capacidad humana de inmersión. No obstante, su desarrollo sufrió una desaceleración debido a la caída del precio del petróleo y a la recesión económica mundial.

Desde entonces los ROVs comenzaron a usarse tanto para las construcciones iniciales de operaciones submarinas como para su consiguiente mantenimiento y reparaciones.

De hecho, el abaratamiento en el diseño de los ROVs ha permitido que su uso se abra a más mercados, no sólo el de las investigaciones científicas, operaciones militares o construcciones submarinas, sino a opciones como el de ocio o la creación de contenidos audiovisuales.

Finalmente, empresas como Mitsui o JAMSTEC han conseguido crear ROVs capaces de explorar lugares tan recónditos como la Fosa de las Marianas, con inmersiones de hasta 10911,4 metros con el ROV Kaiko.

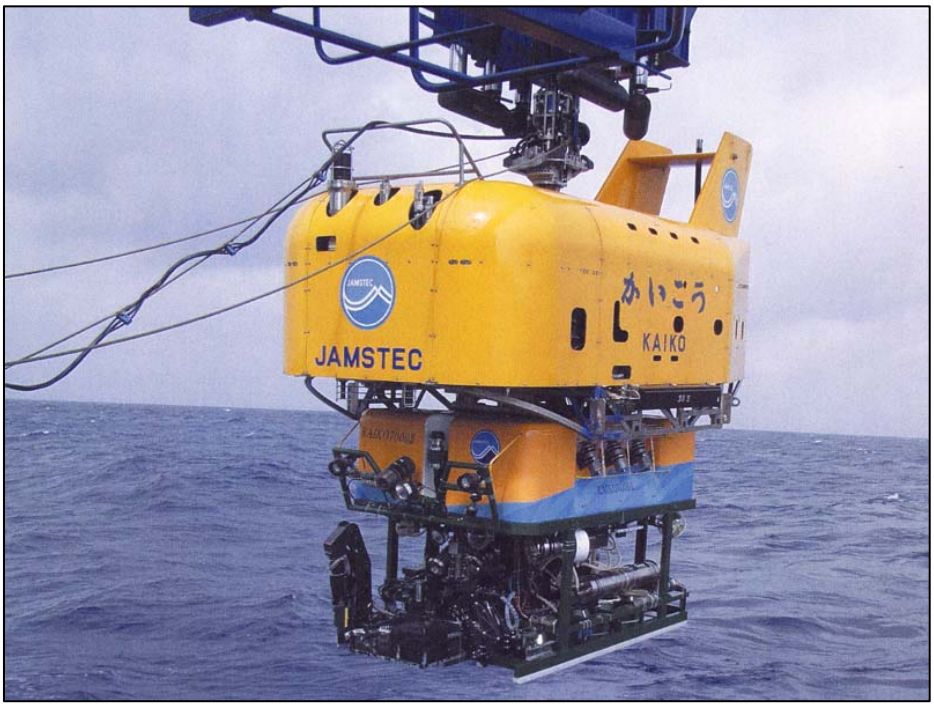


Figura 3. ROV Kaiko desarrollado por la empresa japonesa JAMSTEC[4].

2.2.Categorías y consideraciones de diseño

En esta sección se analizan tanto la clasificación existente en los vehículos ROV atendiendo al tamaño y funcionalidad, como los parámetros que se tienen en cuenta a la hora de diseñar un vehículo submarino operado remotamente.

2.2.1. Categorías

Los ROVs se pueden clasificar atendiendo al tamaño, capacidad de inmersión, potencia, o la naturaleza de sus manipuladores en caso de que los lleven [5][6].

- ROVs de pequeño tamaño (eléctricos). Con un máximo de 15 kg de peso, son usados habitualmente como alternativa a los buceadores. Disponen de una potencia inferior a 5 CV (3,68 kW), y su profundidad operativa máxima se encuentra en torno a los 200m.
  Pueden disponer de pequeños manipuladores robotizados y de sónar.
  Existe una subcategoría, la de microROVs, con pesos inferiores a los 3 kg y cuya función es observar lugares donde un buzo no puede entrar, como el interior de tuberías.
  Este tipo de ROVs normalmente no necesita más de 1 persona para su operación.

Figura 4. QUEST LCROV[7].

- ROVs de tamaño medio (eléctricos/hidráulicos). Normalmente disponen de una potencia menor a 100 CV (73,54 kW), un peso de entre 1000 y 2200 kg y una capacidad de carga de entre 100 y 200kg. Alcanzan profundidades en torno a los 1000m.

Se trata de la categoría más utilizada para tareas de observación y soporte de perforaciones.

Figura 5. ROV Viper de la compañía Perry Tritech[8].

- ROVs de gran tamaño (eléctricos/hidráulicos). Disponen de potencia comprendida entre 100 y 250 CV, con capacidad de elevación de carga de hasta 5000kg. Con pesos que oscilan entre los 2000 y 6500kg, esta clase de ROVs pueden alcanzar profundidades cercanas a los 3000m.
  Son utilizados principalmente para la industria petrolera, por la capacidad de realizar tareas pesadas.

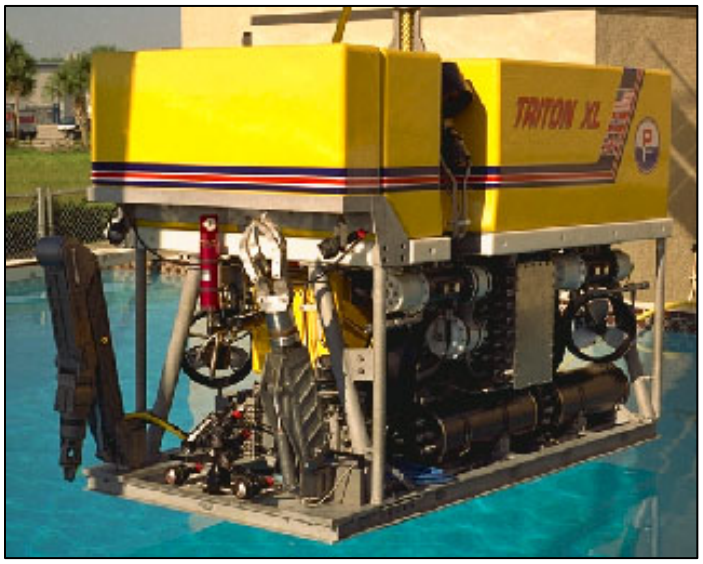


Figura 6. TRITON XL[9].

En esta categoría se debe mencionar la subcategoría de ROVs para grandes profundidades. Estos ROVs disponen de la potencia mínima necesaria para alcanzar profundidades por debajo de los 4000m. Por ejemplo, el ROV KAIKO fue capaz de alcanzar el punto más profundo del fondo marino, en la Fosa de las Marianas (10911,4 m), en el año 1995.

- ROVs arrastrados. Esta categoría abarca un gran número de diferentes vehículos que son remolcados por barcos, principalmente para tareas de investigación y soterramiento de cables de telecomunicaciones.
  Existen dos tipos de vehículos para esta tarea. Los encargados de crear la zanja necesaria para el cable, y los que propiamente lo entierran.
  El diseño de los primeros varía en función del suelo en el que se vaya a emplear, pudiendo llegar a pesos de 80 toneladas y fuerza de arrastre de 240 toneladas.

Figura 7. ROV para abrir zanjas submarinas de la compañía Prysmian[10].

- ROVs autónomos. Son los ROVs que no necesitan un operador en la superficie para su control, por lo que no disponen de cable de alimentación y transmisión de datos.

2.2.2. Consideraciones de diseño

Dado que las características de este tipo de vehículos no son fijas, su proceso de diseño es muy abierto.

El rendimiento final de un ROV consiste en un equilibrio entre el diseño y las características buscadas. El sistema se puede dividir en varios subsistemas, entre los que se encuentran generalmente el chasis, los sensores, el sistema de control y el cable de energía y señal.

La interrelación entre dichos subsistemas se sintetiza gráficamente en la *espiral del diseño de un ROV* [11]:

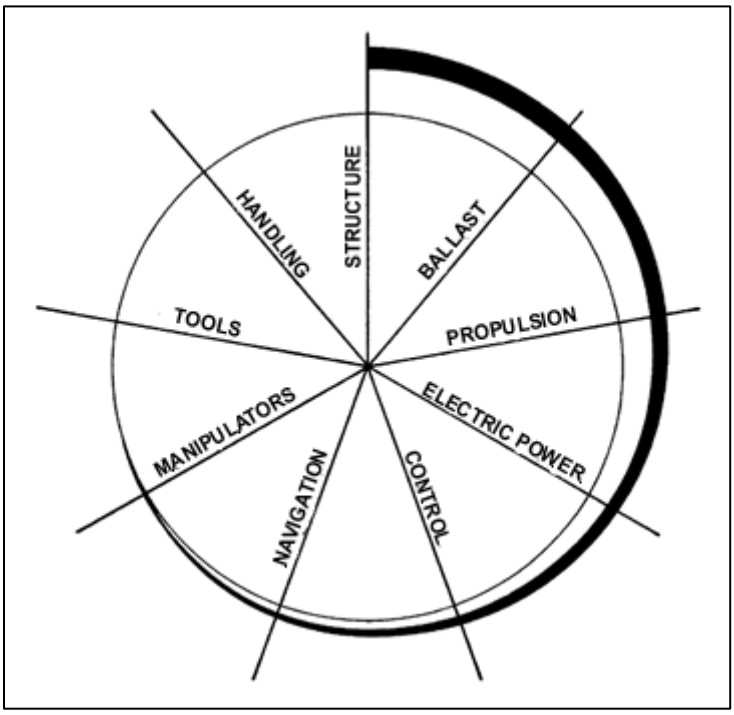


Figura 8. Espiral de diseño de un ROV[12].

Dicho diagrama trata de explicar de una manera visual la relación entre los subsistemas que conforman el ROV.

Por ejemplo, un ligero aumento (en la fase de diseño) de velocidad conlleva un cambio en el sistema de entrega de energía eléctrica para el propulsor. Y este cambio a su vez deriva en un rediseño en la configuración, pues es probable que se deba cambiar la estructura, el peso, el sistema de control, lastres o el tamaño del cable, entre otros.

En resumen, una pequeña modificación puede desencadenar cambios mayores en otros subsistemas.

A continuación se describen brevemente las consideraciones más destacadas a la hora de diseñar un ROV:

- Aspectos generales.

  Los ROVs han de ser vehículos con diseños flexibles ya que su configuración debe adaptarse a distintas operaciones. Los aspectos a tener en cuenta a la hora de diseñar un ROV son el coste total, la plataforma desde donde se va a operar (barco, plataformas), energía necesaria para su funcionamiento, potencia, tamaño, peso, profundidad máxima alcanzable o capacidad de carga, entre otros.

- Arrastre.

  La velocidad que puede alcanzar un ROV depende de la potencia disponible y del arrastre existente. Este último está compuesto por el arrastre que genera el propio vehículo y el que genera el cable:

$$Arrastre = 0'5 * s * A_{veh} * V^2 * Cd_{veh} + 0'5 * s * A_{cable} * V^2 * Cd_{cable}$$

  Donde:

  A_{veh} = área característica, normalmente la sección transversal frontal del ROV.

  A_u = área característica del cable = $\dfrac{\text{diámetro del cable} * \text{longitud del cable}}{12}$

  s = $\dfrac{\text{densidad del agua salada} \left(64\ \frac{lb}{ft^3}\right) o\ (1.025 \frac{kg}{m^3})}{\text{gravedad} \left(32.2\ \frac{ft}{s^2}\right) o\ (9.81\ \frac{m}{s^2})}$

  V = velocidad del vehículo $\left(\frac{ft}{s}\right)$ o $(\frac{\text{m}}{\text{s}})$

  Cd_{veh} = coeficiente de arrastre. Normalmente entre 0.8 y 1.

  Cd_{cable} = coef. de arrastre del cable. Entre 1.2 y 0.2.

Si se personaliza la ecuación para un caso en el que el vehículo se mueva a 1 nudo/h (1.69 ft/s o 1.9 km/h), a 1000 pies de profundidad (305m), con un diámetro del cable de 1 pulgada (2.54 cm) y una superficie frontal del vehículo de 16 ft^2 (1.5 m2) se obtiene:

$$Arrastre\ del\ vehículo = 36\ lb\ (16.3kg)$$

$$Arrastre\ del\ cable = 284lb\ (129kg)$$

Por lo que se demuestra que los cambios geométricos en el ROV no logran un cambio significativo en el conjunto, pues el cable genera normalmente más arrastre.

- Lastres y flotabilidad.

El peso total de un ROV consiste en el peso de los componentes, la carga útil y la flotabilidad requerida para lograr una correcta estabilidad.

Lo más habitual es dotar al vehículo de una flotabilidad positiva, esto es, que haga ascender al vehículo hacia la superficie por sí mismo. Esto permite recuperarlo en caso de un fallo en los propulsores. Dicha flotabilidad oscila normalmente entre los 2.3kg para vehículos de pequeño y 6.8kg para aquellos más grandes.

La medición de estabilidad de un ROV se estima mediante la siguiente ecuación:

$$m = W * BG * sin\theta$$

Donde:

$m =$ $momento\ requerido\ para\ cambiar\ el\ ángulo\ de\ cabeceo\ (pitch)$

$W = peso\ del\ vehículo$

$BG = distancia\ desde\ el\ centro\ de\ flotabilidad\ al\ centro\ de\ gravedad$

$\theta = ángulo\ de\ cabeceo\ (pitch)$

Una distancia BG mayor, la cual se puede conseguir colocando los pesos en la parte inferior del vehículo y la flotabilidad en la superior, consigue un vehículo más estable.

- Cable.

Para el diseño del cable se tiene que tener en cuenta aspectos como el tamaño del vehículo, peso, profundidad máxima operativa o potencia.

El cable también ofrece el soporte mecánico para el ROV, ya que tiene que soportar el peso total y las cargas externas que surjan, como las corrientes de agua.

El acero suele ser el material más empleado como elemento estructural del cable. No obstante, se puede hacer uso de fibras sintéticas como el Kevlar, que reducen el peso a cambio de un coste total más elevado. Para sistemas que operan en profundidades muy acentuadas el uso de las fibras sintéticas es la única opción debido al factor de arrastre que tendría un cable con acero como elemento de refuerzo, que haría prácticamente inoperable el vehículo.

- Propulsión.

Los sistemas de propulsión suelen ser generalmente eléctricos en ROVs con un peso inferior a los 225kg debido a la poca eficiencia y alto peso de los sistemas electro-hidráulicos.

En los sistemas eléctricos se emplea un motor de frecuencia variable para cada propulsor.

La consideración más importante que se ha de tener en cuenta a la hora de diseñar el sistema de propulsión es que el empuje generado por el propulsor disminuye conforme la velocidad aumenta. Por ello se ha de analizar la posición del propulsor cuidadosamente.

- Iluminación.

Las observaciones subacuáticas se dificultan debido a la absorción de la luz por el agua y a las partículas presentes en ella. La materia disuelta aumenta la

absorción, y la suspendida aumenta la difusión. Esta última es la que más problemas plantea porque provoca la iluminación del agua de los alrededores, no sólo de la que está frente al foco.

Otro problema añadido es que la absorción de la luz varía en función del color. La luz roja se absorbe unas 6 veces más rápido que la verde o azul, y es el motivo por el que las fotografías subacuáticas tienen siempre tonos azulados y sin apenas gama cromática. En la gráfica se observa la atenuación de la luz roja frente a la azul o la violeta.

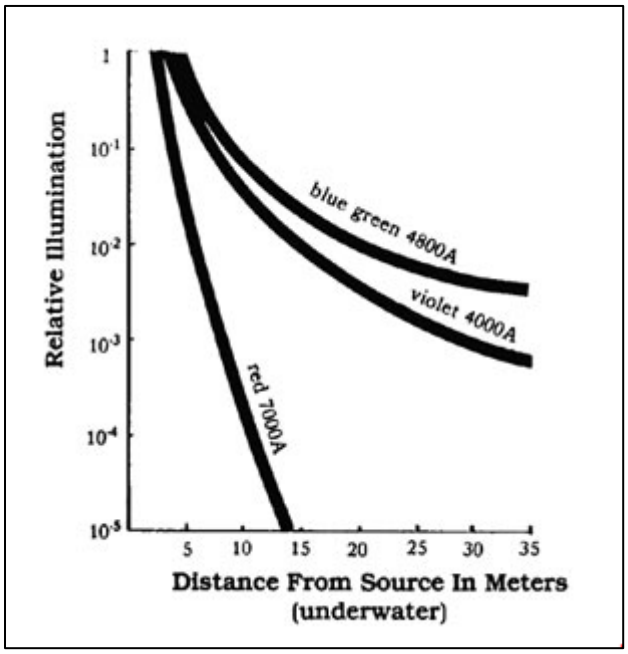


Figura 9. Iluminación relativa frente a la distancia en fuentes de luz monocromáticas[13].

2.3.Estado actual

En la actualidad los ROVs se han convertido en una opción fiable para operaciones tanto en alta mar como en tierra firme (cuevas inundadas o lagos, entre otros) en ámbitos militares, comerciales o científicos.

Lejos de actuar como simples cámaras submarinas, en la actualidad se usan ROVs para tareas como [14]:

- Acompañamiento de buzos para reforzar su seguridad y asistirles en diversas tareas.

- Cartografía de cuevas o zonas inundadas.
- Inspección de plataformas marinas: inspecciones visuales o monitorización mediante instrumentos, para detectar corrosión, grietas o contaminación biológica, incluyendo la detección y sellado de fugas de fuel.
- Inspección de tuberías: como se comentó en el apartado anterior, existen ROVs de tamaño muy contenido capaces de realizar inspecciones en el interior de tuberías para asegurar su correcto estado.
- Estudios visuales y acústicos previos a la instalación de tuberías, cables u otros proyectos marinos.
- Apoyo a la perforación de yacimientos petrolíferos marinos, mediante acciones como la inspección visual y monitorización, o el soporte y mantenimiento de la maquinaria perforadora usando diversos manipuladores robóticos.
- Retirada de escombros y otros deshechos generados en la construcción de instalaciones marítimas.
- Limpieza y mantenimiento de las instalaciones marinas, mediante succionadores, cepillos rotatorios, chorros de agua a alta presión u otros métodos abrasivos para la limpieza.
- Soporte para las telecomunicaciones: desde la inspección para el emplazamiento de nuevos cables submarinos, hasta el propio entierro y mantenimiento de los mismos. Un ejemplo reciente de esta aplicación es la instalación de un nuevo cable de fibra óptica entre España y EEUU con capacidad de transmisión de 160 Terabytes por segundo [15].
- Detección y recuperación de objetos en catástrofes aéreas o hundimientos de embarcaciones.
- Respecto a las aplicaciones científicas, su capacidad para obtener vídeos e imágenes de alta calidad ha permitido a las investigaciones explorar sitios previamente inexplorados.

El primer ROV norteamericano diseñado para tal fin fue el JASON del Instituto Oceanográfico Woods Hope, el cual es capaz de sumergirse hasta 6000m de profundidad. Este vehículo ha completado operaciones científicas como la exploración de navíos hundidos en el Mar Mediterráneo.

Muchos conceptos del JASON fueron adoptados por el Instituto de Investigaciones Acuáticas de Monterey Bay (MBARI) para el *Tiburón*. Este vehículo ha realizado operaciones tales como retirada o depósito de instrumental marino y estudios ecológicos.

En Japón, el Centro Tecnológico y Científico Marino Japonés (JAMSTEC) está desarrollando una serie de ROVs para la recuperación de los sumergibles hundidos *Shinkai.*

Hoy en día, el grueso de los desarrollos científicos se centra en evolucionar los UAVs de bajo coste, es decir, ROVs que funcionan de manera autónoma sin necesidad de un operario a cargo de su control [16].

- Respecto a las aplicaciones militares, la mayoría de las operaciones se centran en tareas de detección de minas, ya que un ROV es mucho más económico que un barco, y evita poner en riesgo la vida de buzos.

  En la actualidad, los avances en los ROVs son evidentes en terreno militar, pues se comienzan a usar para recolección de información en zonas hostiles. Debido a su naturaleza de uso remoto, al igual que los UAV aéreos, los UAV submarinos y ROVs jugarán un papel cada vez más importante en las operaciones militares [17].

En la actualidad, se estima que cerca de 400 ROVs están presentes en proyectos repartidos por todo el mundo, principalmente dedicados a las operaciones petrolíferas. Aquí se exponen brevemente los más destacados [18]:

- Europa: la zona de principal actividad se encuentra en el Mar del Norte en los sectores británico y noruego. La mayoría de estas operaciones se realizan a profundidad media (~150m) o menor. Con ayuda de los ROVs Noruega ha

perforado su pozo más profundo a 1274m, y descubierto un yacimiento petrolífero a 3900m en la Cuenca del Voring.

- Asia y Oceanía: la mayoría de actividades se concentra en la zona comprendida entre el Oeste de Australia hasta Malasia y el Mar del Sur de China. Las compañías petroleras están realizando estudios sísmicos en el Oeste de Australia, a unas profundidades comprendidas entre 900 y 1600m, en búsqueda de depósitos de gas.

- Sudamérica: la mayoría de operaciones que implican el uso de ROVs en esta zona se concentran en Brasil, y más concretamente en la Cuenca de Campos. La petrolera principal de este país, Petrobras, realiza estudios de búsqueda de reservas de petróleo y gas a profundidades cercanas a los 2000m. De hecho, ya tiene proyectado un pozo de extracción a una profundidad de 1835m, el cuál sería el más profundo de la zona.

- Norteamérica: se conocen hasta 104 proyectos de inspección submarina, con más de 100 ROVs realizando tareas de soporte, y 31 perforaciones simultáneas, la mayoría concentradas en el Golfo de Méjico, con profundidades de hasta 3000m.

- Ártico: Rusia es la pionera en la zona, realizando inspecciones en las aguas del Mar de Barents y el Mar de Kara, donde según los estudios podrían encontrarse los mayores depósitos de gas del Ártico [19]:

- África: compañías petroleras como ExxonMobil tienen perforaciones en Nigeria (cercanas a 1500m de profundidad), y exploraciones en el Golfo de Guinea y el Congo.

2.4. Simuladores de robótica submarina

En este capítulo se presentan algunas alternativas a UWSim existentes como simuladores subacuáticos. Se dejan a un lado otros simuladores de robótica que no están especializados en entornos submarinos, como OpenRAVE, OpenHRP3 o Webots.

- Gazebo.

Es un simulador gratuito de robótica para espacios abiertos. Aunque inicialmente las simulaciones submarinas no estaban soportadas, se lanzaron varios complementos (*plugins)* para dotarle de esta funcionalidad y simular el comportamiento de objetos y robots bajo el agua: *BuoyancyPlugin* (que calcula la fuerza de flotabilidad de cada pieza y la aplica al centro de masas) junto a *LiftDragPlugin* [20].

No obstante, no dispone de las funcionalidades completas que dispone UWSim, como el uso de sensores o el sistema de comunicación con ROS para este entorno.

- ROVSim^2 PRO [21].

Se trata de un simulador y entrenador de pilotos para ROVs desarrollado por la empresa Marine Simulation.

Dispone de un diseñador de ROVs, en los que se ajustan los parámetros de propulsión y dinámica, como la potencia y la orientación de los propulsores.

También permite configurar los parámetros del agua, como la dirección y velocidad de las corrientes o la turbidez, que afectan al comportamiento del ROV (por ejemplo, una mayor corriente repercute en el arrastre del cable, y, por tanto, genera más movimientos imprevistos en el ROV).

Como característica más destacada, este simulador incluye un sónar virtual, el cual dispone de un visor similar a un modelo real, en el que se muestra tanto el barrido del sónar como los parámetros de estado del vehículo (profundidad, velocidad o estado de las cámaras, entre otros).

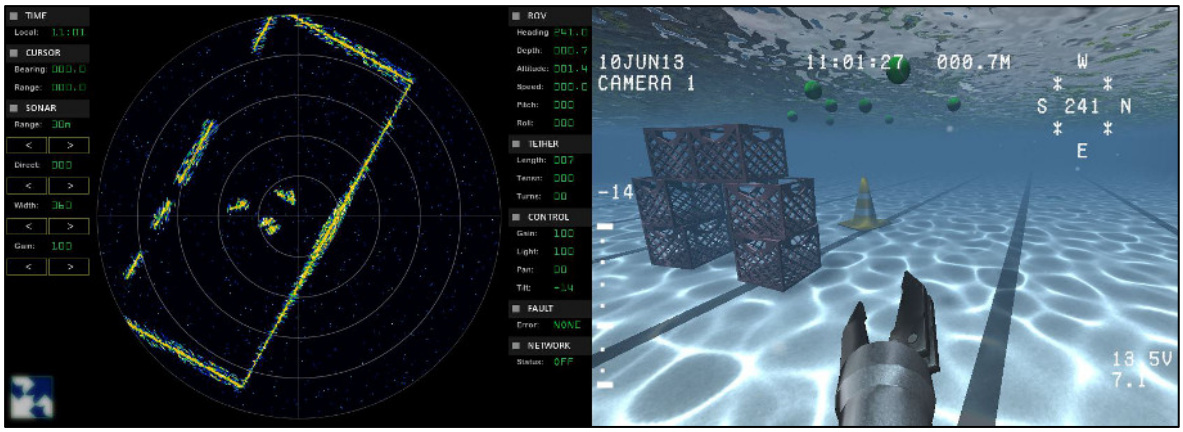


Figura 10. ROVsim2 PRO[22].

- VMAX ROV Simulator [23].

Desarrollado por la empresa española QSTAR SLU, un centro de formación de robótica submarina y de formación de pilotos técnicos de ROVs, VMAX ROV Simulator es un software para desarrollar simulaciones de operaciones con ROVs.

Sus características principales son el gran realismo visual y la fidelidad de las físicas.

El simulador contiene varios escenarios de entrenamiento, como hundimientos y búsqueda de objetos en ruinas sumergidas.

Figura 11. VMAX 2.6 en el escenario de ruinas grecorromanas[24].

- ARI ROV Simulator [25].

Es un simulador que proporciona simulaciones fieles a la realidad para el entrenamiento de pilotos de ROV.

Los vehículos del simulador disponen de 2 brazos robóticos para simular tareas subacuáticas como la recogida de objetos o reparaciones de cables.

El operador dispone de información en tiempo real de los sensores (incluyendo sónar) y cámaras, y puede operar el vehículo a través de distintos mandos, interruptores o selectores.

También permite configurar los parámetros del entorno como la luminosidad y turbidez del agua, o permitir la posibilidad de que aparezcan imprevistos como enredos del cable con objetos del fondo, para aproximar al operador a situaciones reales.

Como característica destacada, permite simular el proceso de recogida y puesta en el agua desde la plataforma que transporta el ROV.

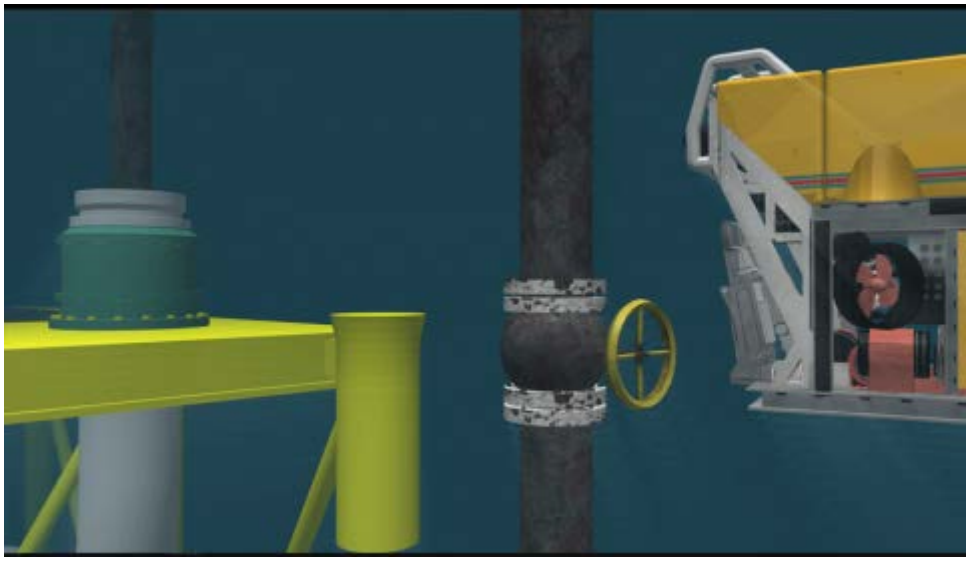


Figura 12. ARI ROV Simulator[26].

3. Herramientas software

La parte técnica de este trabajo se realiza tanto en el Sistema Operativo Ubuntu Linux 14.04.5 LTS 64bits (ROS Indigo y UWSim), como en Microsoft Windows 10 PRO 64bits (Autodesk Fusion 360, Microsoft Office Word, Excel y Power Point).
En orden cronológico se detallan las herramientas software utilizadas:

3.1. Autodesk Fusion 360

Autodesk Fusion 360 es una herramienta software de la empresa estadounidense Autodesk.

Forma parte de la línea de productos Autodesk Inventor, orientada a al desarrollo 2D y 3D de diferentes objetos y escenarios, como distribución de plantas industriales y procesos de producción (Autodesk Factory Design Suite) o diseño final de producto (Autodesk Product Design Suite).

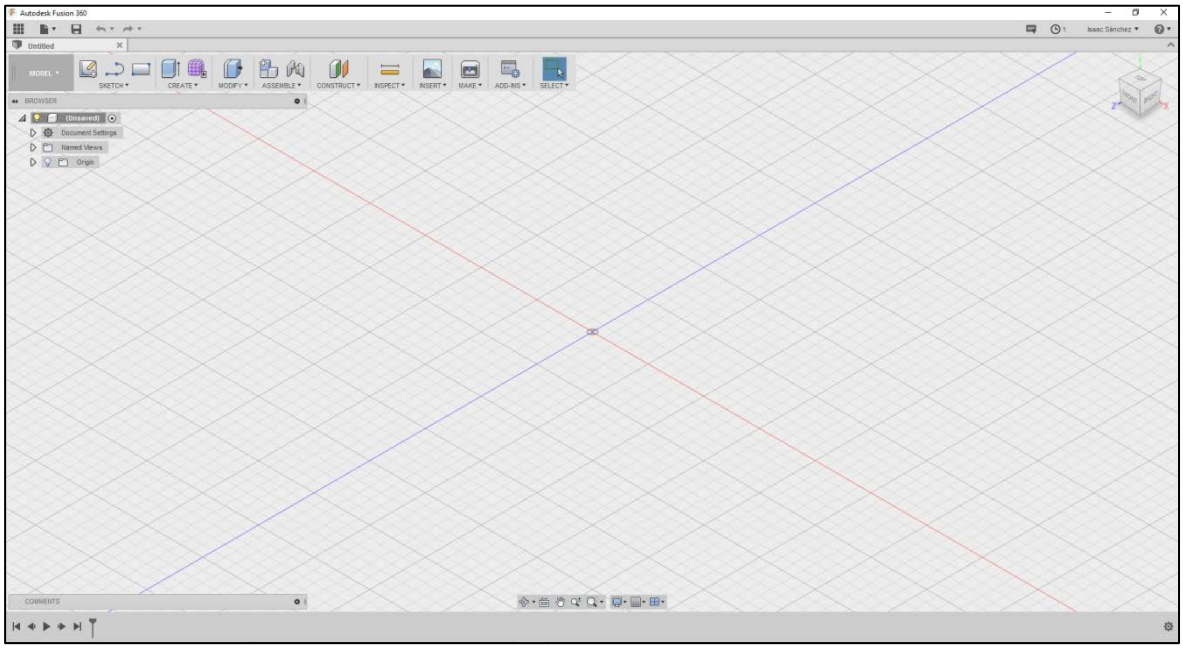


Figura 13. Pantalla principal de Autodesk Fusion 360.

El software Autodesk Fusion 360 combina tanto CAD (*Computer Aided Design*, diseño asistido por ordenador), CAM (*Computer Aided Manufacturing*, fabricación asistida por ordenador) y CAE (*Computer Aided Engineering*, ingeniería asistida por ordenador).

Permite diseñar objetos, examinar movimientos entre piezas, crear conjuntos, y realizar animaciones y renderizados fotorrealistas, así como los flujos de trabajo para manufacturar dichas piezas/conjuntos. Además, emplea una plataforma en la nube mediante la cual pueden guardarse los proyectos, así como permitir la colaboración de varias personas en los mismos desde emplazamientos diferentes [27].

Dispone de licencia gratuita para la comunidad educativa durante 2 años, con todas las características disponibles.

Puede exportar los siguientes formatos [28]:

Sólido:

- F3D - Tipo de archivo de Fusion 360, sólo el tipo de archivo que conserva la información de duración. Los ensamblajes se guardan como archivos F3Z (zip)
- IGES
- SAT
- SMP
- STEP

Malla:

- FBX
- OBJ
- STL

2D:

- DWG
- DXF

Autodesk Fusion 360 puede ser instalado en los sistemas operativos Microsoft Windows y Mac. Por ello se emplea Windows 10 Pro 64 bits en el presente trabajo.

3.2.Ubuntu 14.04.5 LTS

Ubuntu es un Sistema Operativo *open-source* (software libre), con licencia GPL basado en GNU/Linux patrocinado por la compañía británica Canonical, la cual se encarga de mejorarlo con ayuda de desarrolladores de la comunidad.

Dispone de un entorno de escritorio llamado Unity. La comunidad, por su parte, ha desarrollado otros entornos de escritorios como Kubuntu, Ubuntu GNOME o Edubuntu.

Cada 6 meses Canonical se da a conocer una nueva versión de Ubuntu, siendo la más reciente a fecha de realización de este trabajo la versión Ubuntu 17.04 Zesty Zapus.

Cada 2 años se publican las versiones LTS (*long term support)*, como la utilizada para este trabajo, que tienen soporte oficial durante 5 años, por lo que destacan por su estabilidad y actualizaciones [29].

La primera versión de Ubuntu fue la 4.10 *Warty-Warthog*, lanzada en octubre del 2004 con el objetivo de mejorar Debian para hacerlo más sencillo de utilizar para los usuarios.

En la actualidad soporta las plataformas x86, x86-64 (desde sus inicios) y ARM (desde el año 2010 con la versión Ubuntu 10.10, aunque de manera extraoficial, esto es, sin soporte directo de Canonical).

Se decide usar la versión 14.04.5 LTS ya que es la recomendada por los desarrolladores de UWSim.

En el Anexo III se detallan los pasos seguidos para instalar y configurar correctamente este Sistema Operativo.

3.3.ROS

ROS son las siglas de *Robot Operating System*, un espacio de trabajo para crear software orientado al manejo de robots.

Las primeras versiones de ROS surgieron entorno al 2007 para superar una serie de desafíos presentes en el desarrollo de grandes robots de servicios en los programas STAIR (*Stanford AI Robot*) de la Universidad de Stanford, EEUU, y el PRP (*Personal Robots Program*) de Willow Garage, una incubadora de empresas dedicada a la creación de software *open source* (código abierto) en el campo de la robótica [30].

Actualmente ROS cuenta con cientos de desarrolladores e instituciones que usan y mejoran su código para sus propias investigaciones y desarrollos, desde aplicaciones a pequeña escala hasta grandes sistemas industriales automáticos.

Múltiples versiones han aparecido desde su origen, entre las que destacan recientemente Fuerte (mayo 2012), Groovy (diciembre 2012), Hydro Medusa (septiembre 2013), Indigo Igloo (julio 2014), Jade Turtle (mayo 2015), Kinetic Kame (mayo 2016) y Lunar Loggerhead (mayo 2017, la más reciente disponible a la fecha de redacción de este trabajo).

La versión utilizada en este trabajo es Indigo, pues es la recomendada por los responsables de este proyecto en la UC3M Leganés. Esta versión fue principalmente lanzada para Ubuntu 14.04 LTS, de ahí que se haya recomendado esta versión del sistema operativo Linux en detrimento de versiones más recientes.

Las metas de ROS [31] se pueden resumir en:

- P2P (*Peer-to-Peer*): un sistema construido en ROS consiste en un numero de procesos, con un potencial número de servidores, conectados simultáneamente en una topología P2P. Los tipos de comunicación en ROS son numerosos incluyendo comunicaciones síncronas RPC como *Services*, el paso de mensajes asíncronos como *Topics*, y un sistema de almacenamiento de datos que en ROS se conoce como el servidor de parámetros, *Parameter Server*.

La topología P2P requiere un árbitro, u organizador de conexiones, que permita a distintos procesos sincronizarse durante la comunicación simultanea de todos los procesos. Dicho árbitro recibe el nombre de *service* o *master*.

- Multilenguaje: ROS soporta 4 lenguajes de programación diferentes: C++, Python, Octave y LISP.

  Para soportar el desarrollo con múltiples lenguajes, ROS utiliza un sencillo lenguaje, *Interface Definition Language IDL*, para describir los mensajes enviados entre los distintos módulos del sistema. El IDL utiliza archivos de texto muy compactos para describir los campos de cada mensaje, permitiendo la composición del mensaje original:

```
Header header
Point32[] pts
ChannelFloat32[] chan
```

Figura 14.Mensaje tipo IDL.

  Esto permite reducir considerablemente el tiempo empleado y los errores del programador: las 3 líneas de código de la figura previa se expanden automáticamente en un archivo de 137 líneas de código en lenguaje C++, 96 en Python, 81 en Lisp y 99 en Octave.

- Basado en herramientas: para manejar la complejidad de ROS, se opta por un diseño basado en microkernel, donde un gran número de pequeñas herramientas se usan para crear y ejecutar numerosos componentes de ROS. Dichas herramientas realizan acciones variadas como navegar por la estructura del código, definir parámetros de configuración, crear una interfaz visual de la conexión P2P, medir el ancho de banda utilizado, generar documentación de manera automática, entre muchas otras.

- Sencillo de replicar: todos los drivers y desarrollos de algoritmos tienen lugar en librerías independientes a ROS. El sistema de desarrollo de ROS

crea desarrollos modulares dentro de la estructura en árbol del código utilizando CMake. De esta manera, relegando la dificultad a las librerías, y creando únicamente pequeños ejecutables que las conecten directamente con ROS, se consigue que el hecho de reutilizar código de terceros sea una tarea más sencilla.

ROS reutiliza código de otros proyectos *open-source* como OpenCV (para algoritmos de visión artificial) u OpenRAVE (para algoritmos de planificación).

- Gratuito y *open-source*: mientras que otros espacios de trabajo propietarios como Microsoft Robotics Studio o Webots son únicamente utilizables bajo licencia, ROS se distribuye bajo licencia BSD, que permite utilizarlo para desarrollo de proyectos tanto comerciales como no comerciales.

Elementos básicos de ROS [31]:

- Nodos: son módulos software que realizan la computación. ROS está diseñado para ser modular, por lo que generalmente un sistema está formado por numerosos nodos.
  Cuando múltiples nodos se están ejecutando simultáneamente es conveniente utilizar comunicación P2P, pudiendo obtener una representación gráfica de la misma, en la que los nodos son representados entre óvalos, y la comunicación P2P representa las uniones entre nodos.
- Los nodos se comunican entre ellos a través de mensajes: un mensaje es una estructura de datos escrita en tipos de datos estándar (booleanos, enteros...).
- Un nodo envía un mensaje publicándolo en un *topic* (tópico): el nodo interesado en dicho mensaje se suscribirá al tópico oportuno. Puede haber múltiples publicadores y suscriptores en un mismo tópico, del mismo modo que un nodo puede publicar/suscribirse a varios tópicos a la vez.

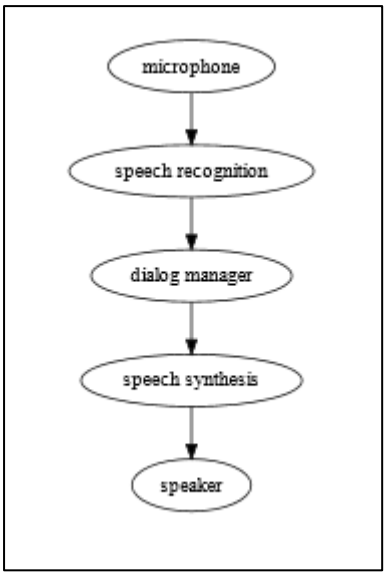


Figura 15. Ejemplo de comunicación en ROS

- Servicios: la comunicación suscriptor/receptor puede ser muy útil en la mayoría de los casos, pero no resulta la más adecuada cuando se trata de interacciones tipo petición/respuesta, utilizadas en sistemas distribuidos. Esta comunicación se realiza mediante servicios, que se definen como un par de estructuras de mensajes, una para realizar la petición y otra para la respuesta. Un nodo ofrece un servicio bajo un nombre determinado y el cliente usa el servicio enviando un mensaje de petición y esperando una respuesta.

El hecho de que los nodos se puedan conocer entre sí durante la ejecución del sistema permite que la gráfica de computación pueda modificada dinámicamente. Esto conlleva un gran aumento de la productividad cuando la complejidad del sistema robótico aumenta. Este hecho es una de las características más potentes y fundamentales de ROS frente a otros espacios de trabajo como previamente mencionados.

A pesar de que cada nodo se puede ejecutar desde la terminal de comandos, introducir dichos comandos para ejecutar un proceso completo puede ser complicado y alargarse en el tiempo si el sistema es complejo.

Para ello, y para facilitar el desarrollo colaborativo, ROS está organizado en paquetes. Un paquete es un directorio que contiene un archivo XML en el cuál se

describen todas las dependencias entre los integrantes del paquete. ROS dispone de una herramienta denominada *roslaunch*, el cual lee la información contenida en un archivo con extensión .XML y ejecuta los diferentes nodos, tópicos y parámetros necesarios. Con esta utilidad, el trabajo del usuario se reduce a introducir una única línea de comando: *roslaunch paquete.xml* (sustituyendo *paquete* por el nombre del paquete a ejecutar).

Dichos paquetes se agregan en repositorios. La herramienta *rospack* fue diseñada para permitir desarrollos simultáneos usando varios repositorios de paquetes.

Finalmente, se ha de mencionar uno de los complementos más potentes de los que dispone ROS: Rviz.

Rviz es un programa distribuido junto a ROS que muestra mediante una interfaz gráfica distintas imágenes, nubes de puntos, gráficos o datos a elección del usuario.

Para la utilización de ROS es necesario crear un espacio de trabajo *catkin_ws* en la carpeta personal del Sistema Operativo. Este procedimiento se explica en detalle en el Anexo VI, mientras que la instalación de ROS Indigo se detalla en el Anexo IV.

3.4.UWSim 1.4

UWSim, *Underwater Simulator*, es un simulador marino para investigaciones y desarrollo robóticos en el ámbito subacuático creado por los investigadores Mario Prats y Javier Pérez del IRSLab de la Universidad Jaume-I de Castellón, los cuales están al cargo también de su mantenimiento.

Permite probar e integrar algoritmos de percepción y control a los proyectos de investigación antes de probarlos en vehículos submarinos reales [32].

UWSim comenzó como herramienta para probar e integrar algoritmos de control y percepción antes de incorporarlos en los robots de los proyectos RAUVI y TRIDENT [33].

Visualiza un escenario virtual submarino que puede ser configurado mediante software de modelado 3D como Blender. A dicho escenario se le puede añadir vehículos subacuáticos, brazos robóticos o sensores simulados.

El simulador está desarrollado en lenguaje de programación C++, y hace uso de las librerías OSG (*OpenSceneGraph)* y osgOcean.

OSG se trata de un proyecto *opensource* (código abierto) de programación para aplicaciones gráficas 3D utilizado por los investigadores para tareas como videojuegos, realidad virtual modelado y visualizaciones científicas. Está desarrollado en código C++ y hace uso de la librería OpenGL.

osgOcean se trata de otro proyecto *opensource* desarrollado como parte del proyecto de investigación VENUS que crea renderizados subacuáticos realistas utilizando OSG.

UWSim ha sido desarrollado buscando los siguientes objetivos:

- Ser fácilmente integrable con arquitecturas de control existentes. En numerosos casos el simulador actúa como un mero visualizador del resultado de algoritmos externos.
- Ser fácilmente escalable, es decir, sencillo para añadir nuevos vehículos, manipuladores robóticos o sensores. Además, se añade soporte para *widgets* (se explicará a continuación).

- Incluir soporte para manipuladores subacuáticos, permitiendo así simular misiones de intervención submarina.
- Ser visualmente realista, y permitir la configuración de diferentes parámetros como el color del agua, visibilidad o cantidad de partículas en suspensión.

Las características principales de este simulador submarino son [34]:

- Entorno configurable: elementos como vehículos, texturas o sensores pueden añadirse, modificarse o eliminarse dinámicamente del sistema. OSG representa el escenario virtual con un gráfico de dicha escena, donde los nodos pueden ser gestionados.

  Otros nodos configurables, como la visibilidad subacuática, el color del agua o las partículas en suspensión, pueden añadirse automáticamente para representar dichos efectos en la escena.

  La escena se describe mediante un archivo XML o un xacro, que es un lenguaje macro para crear archivo XML de una manera más automatizada.

  Además, la escena se puede exportar en los formatos admitidos por OSG, esto es, .ive, .3ds o .wrl.
- Soporte a múltiples vehículos: un vehículo es un archivo 3D que puede ser posicionado en la escena con 6 grados de libertad (X Y Z R P Y). Dichos vehículos se describen mediante un archivo XML de acuerdo al formato URDF. Dicho formato incluye la información relativa a la cinemática, dinámica y aspecto visual del vehículo.

  Múltiples robots pueden ser gestionados de manera simultánea por el simulador.
- Sensores y actuadores: 12 de ellos están disponibles para los vehículos, además de los sensores de velocidad y posición:
  - Cámara: genera imágenes virtuales usadas, entre otros casos, para el desarrollo de algoritmos de visión artificial.
  - Cámara de alcance: imagen de profundidad de la cámara.
  - Sensor de alcance: mide la distancia sobre el objeto cercano.
  - Recolector: simula el agarre de un objeto cuando este está a una distancia predefinida.

  - Sensor de presión.
  - DVL: estima la velocidad lineal del vehículo.
  - Imu: estima la orientación del vehículo con respecto a los ejes de coordenadas del entorno.
  - GPS: indica la posición del vehículo únicamente cuando este está próximo a la superficie del agua (debido a la dificultad de las ondas de radio para propagarse en este medio).
  - Sensor multihaz: simula un conjunto de sensores de distancia que mide múltiples distancias hasta un objeto en el eje Z variando la rotación en el eje X en incrementos angulares desde un ángulo inicial hasta uno final, siendo los 3 parámetros configurados por el usuario [35].
  - Sensor de fuerza y momento sobre una pieza del vehículo.
  - Proyector láser o de luz.
  - Draga: simula dragar el barro/arena del fondo marino para desenterrar objetos.
- Físicas de colisiones: se logra integrando el motor de físicas Bullet con OSG mediante osgBullet. Esto permite simular contactos con distintos grados de fuerzas con las piezas, adaptando la escena automáticamente. El cambio en la forma de una pieza debido a una colisión se genera automáticamente con los modelos 3D de dicha pieza.
- Uso en red: los sensores y actuadores pueden ser controlados por software de terceros a través de la red. Por ello se integra UWSim con ROS, puesto que este último presenta numerosas ventajas para comunicaciones y computación distribuida.
- *Widgets*: son pequeñas ventanas que se colocan en el visor principal del simulador, y permiten mostrar datos concretos elegidos por el usuario (por ejemplo, las imágenes de una cámara, distancia de un determinado sensor o la posición de una determinada articulación).

La estructura de carpetas que utiliza UWSim se detalla a continuación, indicando la ubicación de los archivos más importantes para la realización de este trabajo:

En primer lugar, se trabaja en el espacio de trabajo *catkin* creado (ver Anexo VI para una explicación detallada). Dentro del espacio de trabajo, la carpeta que contiene la mayoría de archivos de código del UWSim se denomina *src*. En ella se distinguen los siguientes elementos:

Figura 16.Carpetas existentes en la carpeta src.

Las carpetas *uwsim_bullet*, *uwsim_osgbullet*, *uwsim_osgocean*, *uwsim_osgworks y visualization_osg* corresponden a librerías de UWSim.

En el directorio \underwater_simulation\uwsim\data\scenes se ubicarán los 2 archivos configurables más importantes para el interés de este trabajo: bluerov2.urdf y bluerov2scene.xml.

Bluervo2.urdf: URDF (*Universal Robotic Description Format*) es un archivo XML usado en ROS para describir todos los elementos de los que se compone un robot (partes y articulaciones). En el Capítulo 6.2 se detalla la estructura del usado para este trabajo.

Bluerov2scene.xml: se trata de la escena de la simulación. Una escena es un archivo XML dividido en varios bloques que definen diferentes aspectos de la simulación, desde los vehículos y objetos hasta parámetros de la simulación. En el Capítulo 6.3 se desarrolla el archivo de escena creado para este trabajo.

Un tercer archivo importante presente en la carpeta *scenes* es *installScene*. Este archivo se trata de un script que descarga diferentes escenas del servidor del IRSLab y las instala automáticamente en UWSim. Si se decide publicar una escena propia en dicho servidor, estará a disposición de la comunidad para instalarla usando en anterior script.

Otra ubicación necesaria es aquella que almacena los modelos 3D de los vehículos, texturas y otros objetos. Se encuentra en la carpeta raíz del sistema (fuera de *catkin*) y generalmente oculta al usuario.

Para visualizarla basta con seleccionar la opción *view* → *view hidden files* del desplegable de opciones del explorador de archivos.

La carpeta se llama .uwsim. En el directorio .uwsim\data aparecen los siguientes elementos:

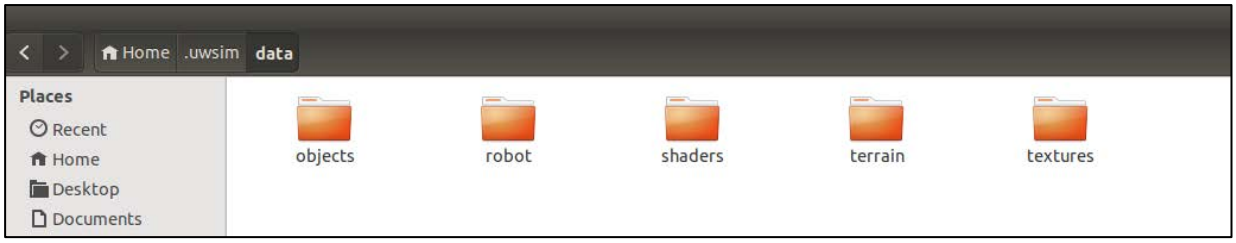


Figura 17. Carpetas existentes en la carpeta data.

Es en la carpeta *robot* donde se han de emplazar los archivos 3D (en formato .osg) del bluerov2. El resto de carpetas permanecerán inalteradas, pues así se aprovecha el terreno y objetos presentes en la versión de demostración de UWSim.

4. Legislación aplicable a vehículos submarinos operados remotamente [36]

Los avances científicos y tecnológicos de la ingeniería marina, civil y militar han permitido el desarrollo y auge de la industria de los denominados drones submarinos o UUV (*Unmanned Underwater Vehicles*), unos vehículos capaces de operar bajo el agua y, a veces, en superficie.

A diferencia de sus semejantes aéreos, las aeronaves civiles pilotadas por control remoto, que han sido objeto de una regulación específica (vid. el Real Decreto-ley 8/2014, de 4 de julio, de aprobación de medidas urgentes para el crecimiento, la competitividad y la eficiencia) ocupando muchos titulares en los medios de comunicación. Por el contrario, los drones marinos y submarinos no cuentan con una regulación propia, habiendo pasado más desapercibidos en los medios hasta ahora.

Todo ello sucede, a pesar de estar cada vez más extendido el uso de los drones y robots marinos en tareas como la cartografía, el peritaje, la vigilancia y salvamento marítimos, la prevención y lucha contra la contaminación marina (incluyendo la detección y el sellado de fugas de fuel), la seguridad, defensa y otras actividades oceanográficas, y de existir en España una creciente industria de ingeniería trabajando en el diseño y construcción de estos prototipos de drones.

Resulta extremadamente difícil establecer un marco jurídico unitario, para unos vehículos que difieren entre sí en características, diseños, dimensiones y uso. Así, por un lado, están los conocidos como ROV (*Remotely-operated Underwater Vehicles*), que son operados a distancia por un humano, y, por otro lado, los AUV (*Autonomous Underwater Vehicles*), robots que operan de forma autónoma. También existen clasificaciones que distinguen entre las denominados MOU (*Mobile Offshore Units*) o unidades móviles offshore, y los UMS (*Unmanned Maritime System*) o vehículos marítimos no tripulados, que en determinados casos permiten transportar a personas y cosas. Finalmente, se aprecian diferencias entre estos vehículos atendiendo a su capacidad de desplazamiento, volumen, diseño y flotabilidad.

Desde una perspectiva legal, y ante la variedad de características que pueden presentar estos vehículos, no sorprende que el estatuto jurídico de estos vehículos no esté claramente definido. Sujetos a la actual Ley 14/2014, de 24 de julio, de Navegación Marítima, muchos de estos drones no tienen un encaje cómodo ni en la definición de "buque" (que requeriría, entre otras cuestiones, que tuviesen capacidad para navegar por el mar y para transportar personas o cosas); ni en el concepto de "artefacto naval", que exigiría que el robot quedase situado en un punto fijo de las aguas.

Asimismo, aunque algunos de estos robots se comportan como boyas subacuáticas, y permanecen estáticos en una zona fija recopilando información, tampoco pueden subsumirse en el concepto de "plataforma fija" utilizado por la Ley de Navegación Marítima, al no encontrarse permanentemente sujetas al fondo de las aguas.

Sin embargo, y pesar de no estar expresamente regulados en la norma citada, estos vehículos operan en el mar, y con ello, están expuestos a riesgos derivados de su actividad marítima (abordajes, averías, etc.), ya esté relacionada o no con la navegación propiamente dicha, y pueden ser objeto de analogías jurídicas propias de los buques y artefactos navales, tales como los contratos de construcción y compraventa, o el régimen de la responsabilidad civil derivada de daños causados por contaminación marina.

Todo ello hace plantear la necesidad de regular expresamente el uso de estos vehículos, especialmente ante la previsión de crecimiento de esta industria, que por otro lado contribuye al afianzamiento del I+D+I español, pudiendo generar puestos de trabajo y atraer inversión.

Desde el punto de vista de los riesgos, y a pesar de que la Ley de Navegación Marítima no incluye expresamente a este tipo de vehículos dentro de los intereses asegurables objeto del seguro marítimo, (como sí hace, por el contrario, con los buques y artefactos navales, incluso en construcción o desguace). La propia ley prevé, que pueden ser objeto de seguro marítimo cualesquiera intereses patrimoniales legítimos expuestos a los riesgos de la navegación marítima, lo que incluiría a los UUV, o al menos, a aquellos UUV capaces de desplazarse y "navegar" de forma autónoma o por control remoto.

En la práctica, el mercado asegurador del Lloyds londinense (mercado de seguro marítimo predominante a nivel mundial) lleva años empleando pólizas específicamente adaptadas para cubrir el equipo marino, submarino y offshore, que incluye los ROV y AUV, proporcionando coberturas a estos vehículos; tanto frente a los riesgos propios de su actividad marítima, como también a los riesgos derivados de su exposición, transporte terrestre, almacenamiento o manipulación a bordo de otros buques y/o en puertos.

Además de las coberturas propias de daños materiales al vehículo (*physical damages*), las pólizas de seguro de los UUV también prevén coberturas específicas relacionadas con la responsabilidad civil frente a terceros por daños causados por este tipo de vehículos, así como coberturas adicionales por pérdidas económicas o lucro cesante (*consequential damages*).

En definitiva, se trata de un mercado muy específico, en el que es recomendable que los operadores de este tipo de vehículos cuenten con agentes, *brokers* y asesores especializados, que conozcan las particularidades de la actividad y las características del UUV de cara a representar los intereses del operador y obtener una adecuada cobertura aseguradora a un coste razonable.

De esta forma se concluye que, con el auge progresivo de la industria de los drones marinos y submarinos, surgen una serie de retos legales y regulatorios, así como muchas oportunidades para un sector marítimo que, a buen seguro, acabará ocupando tantos titulares en los medios de comunicación como ya sucede con sus "parientes", los drones aéreos.

5.BlueROV2

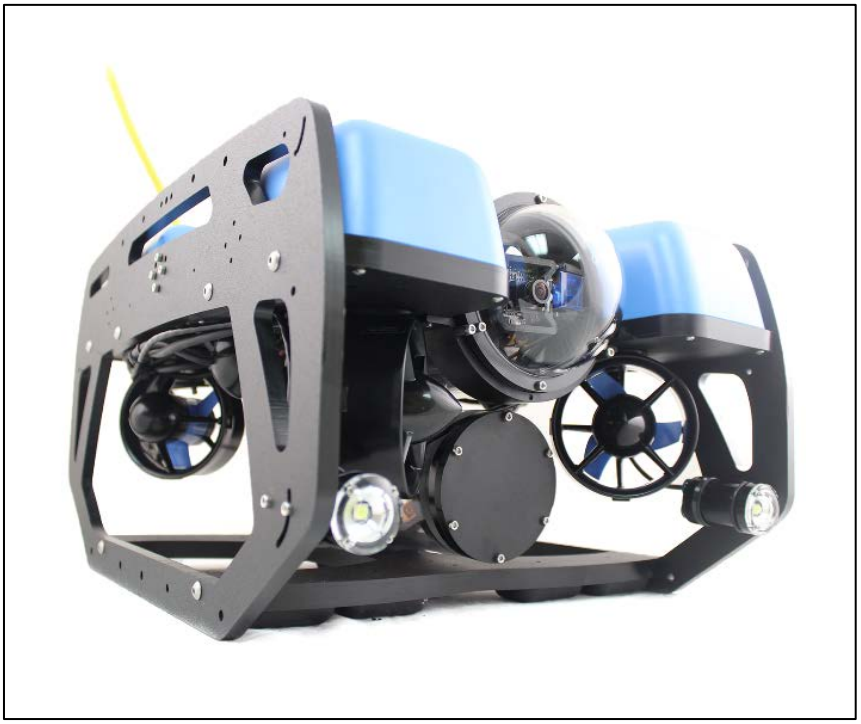


Figura 18. BlueROV2 vista fronto-lateral[37].

El vehículo protagonista del proyecto que lleva a cabo el Departamento de Sistemas y Automática de la Universidad Carlos III de Madrid, y, por ende, el de este trabajo, es el BlueROV2 [38], creación de la empresa estadounidense Blue Robotics.

El BlueROV2 es la evolución del BlueROV, primer modelo de la compañía.

Tiene como propósitos principales las investigaciones submarinas, así como la exploración por ocio, con profundidades recomendadas de hasta 100m (aunque se han realizado pruebas con una profundidad de 130m). Dispone de cable de comunicación de hasta 300m de longitud.

El BlueROv2 utiliza un software de código libre llamado *ArduSub*, y un piloto automático *PixHawk* que le otorga capacidad autónoma. Ambos software son periódicamente actualizados y mejorados por BlueRobotics para mejorar las capacidades del vehículo.

Las características del BlueROV2 resumidas se detallan a continuación (disponiendo de un datasheet con todas las características en el Anexo II):

- Retransmisión en directo mediante vídeo de baja latencia con calidad FullHD 1080p.
- Configuración de propulsor vectorial altamente maniobrable.
- Estable y optimizado para misiones de inspección e investigación.
- Interfaz de usuario multiplataforma y fácil de usar.
- Expandible, ya que cuenta con 3 orificios libres para cables.
- Propulsores T200 (desde 0.01 kgf hasta 5.1kgf de empuje, 300/3800 rpm y un peso de 344g.
- Capaz de realizar inmersiones de 100m, con hasta 300m de cable disponible.
- Energía proporcionada por baterías de sustitución rápida.
- Software de control (*ArduSub*) y hardware de código abierto.

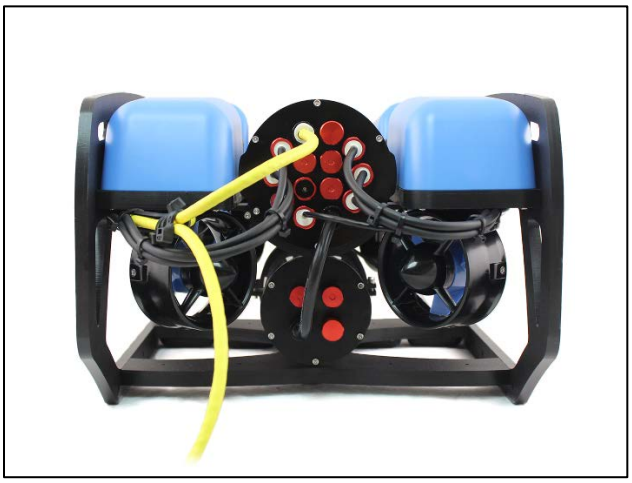


Figura 19. BlueROV2 vista trasera[37].

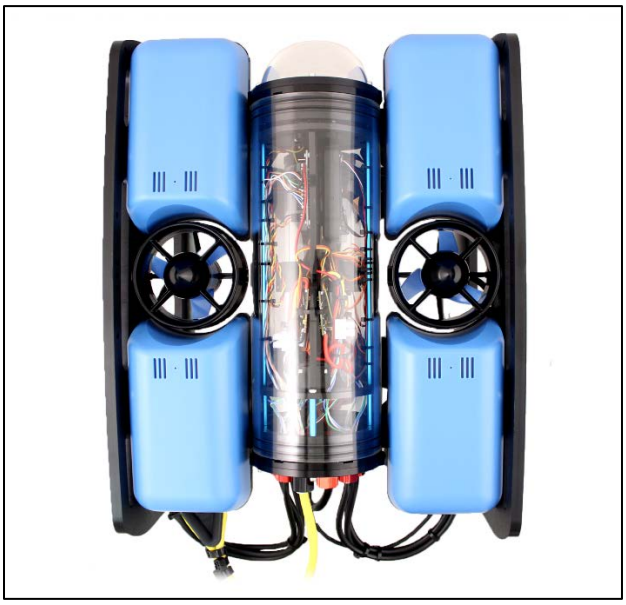


Figura 20. BlueROV2 vista superior[37].

Figura 21. BlueROV2 vista inferior[37].

6. Modelado del robot submarino

Como ya se detalló anteriormente, el presente trabajo tiene como objetivo poder integrar el robot BlueROV2 en el entorno de simulación subacuática UWSim. Para ello, es necesario realizar 3 tareas principales:

1. Exportar las piezas 3D necesarias del BlueROV2.
2. Importar dichas piezas en la estructura interna de UWSim, creando el archivo URDF requerido para el simulado.
3. Crear la escena y la interfaz de comunicación de ROS entre sensores y actuadores para poder realizar análisis y controlar el vehículo.

6.1. Exportación y ajuste del modelo 3D

La primera etapa del trabajo es obtener las diferentes piezas que conforman el robot BlueROV2 en un formato aceptado por el simulador UWSim.

En este caso, atendiendo a los archivos de demostración que se encuentran en las carpetas internas del UWSim, *cirs.xml y g500.urdf* (entre otros) se observa que los archivos 3D del robot de demostración (Girona500) tienen la extensión .osg.

No obstante, se observa y comprueba que UWSim también es capaz de trabajar con archivos con extensión .stl, por lo que, dado que este último es un archivo ampliamente utilizado en herramientas 3D, se decide usarlo para el trabajo.

Como se detalló anteriormente, para importar las piezas se utiliza el software 3D Autodesk Fusion 360.

En este caso, se observa que no dispone de la opción de importar archivos directamente en formato .osg, pero sí en .stl, entre otros, lo cual no es un problema para continuar con el desarrollo.

De todos modos, en caso de querer usar archivos con extensión .osg en el simulador, la limitación de exportación del Autodesk Fusion 360 se solventaría de

una manera sencilla, pues se importarán en .stl y posteriormente se cambiará el formato de dichos archivos en Linux vía terminal, usando el comando:

```
isaac@isaac:~$ osgconv archivo_inicial.stl archivo_final.osg
```

Esta orden crea un nuevo archivo con extensión .osg manteniendo el archivo .stl original.

El archivo 3D del conjunto BlueROV2 se obtiene mediante descarga gratuita en la web GrabCad en formato .rar [39].

Al abrir el conjunto por primera vez con Autodesk Fusion 360 se observa el BlueROV2:

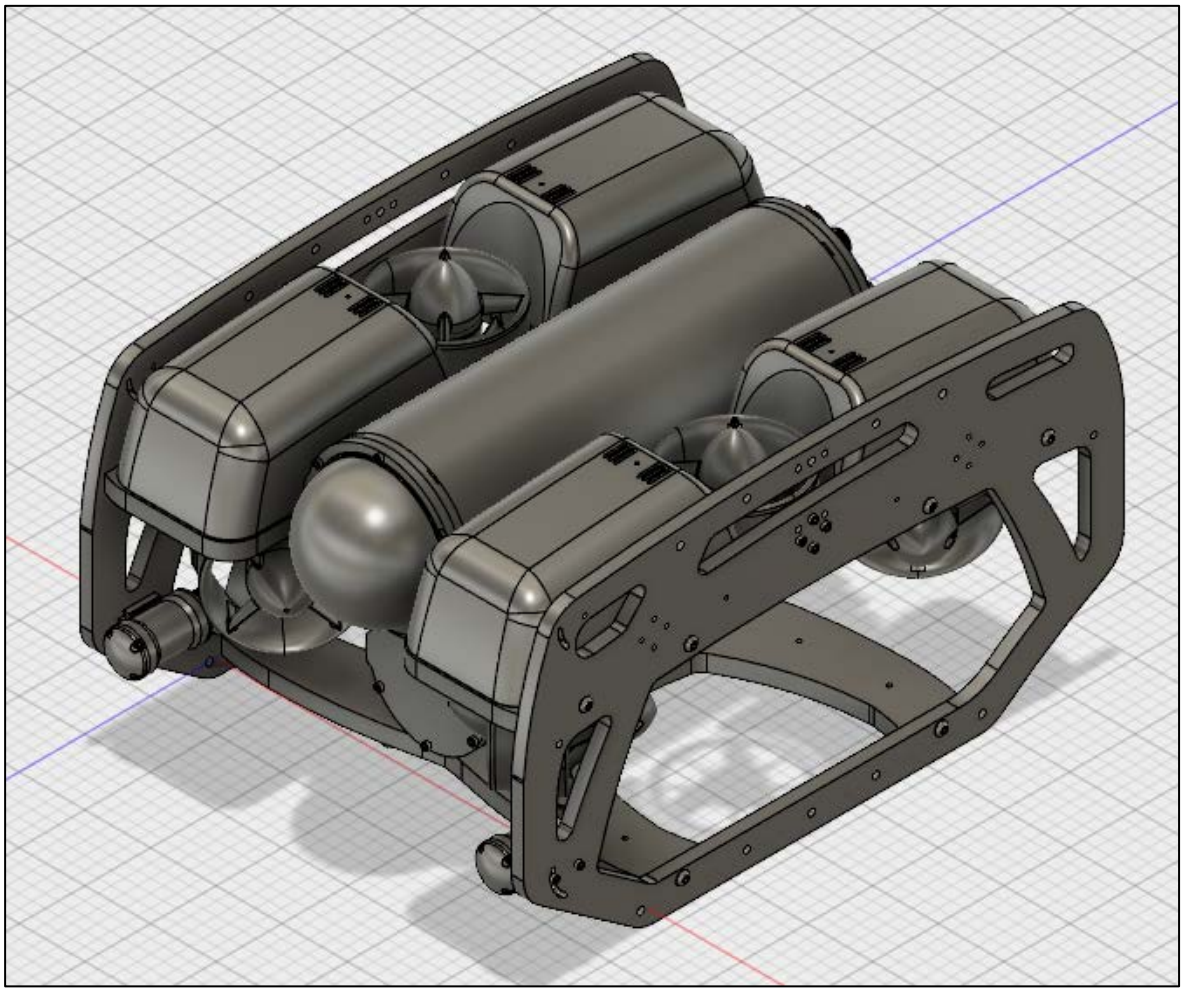


Figura 22. BlueROV2 en Autodesk Fusion 360.

NOTA: a la hora de abrir el archivo .stl descargado del repositorio de GrabCad, Autodesk interpreta que todos los componentes son de metal, por lo que no se

detallan los pesos de cada pieza por separado, ya que son completamente erróneos. Como ejemplo, la espuma de flotabilidad obtiene un peso de 4.002 kg por pieza en Autodesk, cuando en el datasheet oficial se observa una flotabilidad total (4 piezas de espuma, sin contar lastres) de 1.4 kgf. Esto se revierte aplicando un cambio de materiales a varias piezas, consiguiendo un peso cercano al real, pero no exacto.

De igual manera el vehículo no dispone de colores en la figura anterior ni en el trabajo debido al origen del modelo 3D.

La lista de componentes que forman parte del conjunto original es la siguiente:

NOMBRE	DEFINICIÓN	CANTIDAD
PENETRATOR-M-NUT-10-A-R2_94036A625	Tornillo para conexionado de cables	13
BROV-M-BATTERY-CRADLE-R1	Soportes inferior y superior del cilindro de la batería	2
WTE3-M-END-CAP-R1	Tapa frontal del cilindro de la batería	1
WTE3-M-FLANGE-SEAL-R1	Junta frontal de estanqueidad del cilindro de la batería	2
WTE3-P-TUBE-SHORT-R1	Carcasa cilíndrica que contiene la batería	1
WTE3-M-END-CAP-4X10MM-R2	Junta trasera de estanqueidad del cilindro de la batería	1
PENETRATOR-M-BOLT-BLANK-10-25-R2_94036A625	Tornillo para conexionado de cables de batería	2
VENT-M-BOLT-10-25-A-R3_94036A625	Tornillo para conexionado de cables de batería	1
VENT-M-PLUG-10-A-X2A	Tornillo de estanqueidad	1
VBROV-P-BOTTOMPANEL-R1	Panel inferior del chasis	1
VBROV-P-SIDE-PANEL-R1	Panel lateral del chasis	2
Assem-front-panel-assembly	Panel horizontal frontal del chasis	1
Assem-rear-panel-assembly	Panel horizontal trasero del chasis	1
M5x0	Tuerca M5	4
M4x14-socket-head-cap-screw-X1A_92290A150	Tornillo M4x14mm	4

M3x12-socket-head-cap-screw-X1A_92290A117	Tornillo M3x12mm	11
M5x16-button-head-cap-screw-X1a_94500A232	Tornillo M5x16mm	12
Assem-electronics-enclosure	Cilindro de la electrónica (incluida)	1
VBROV-P-BUOYANCY-FOAM-X1A	Espuma de flotabilidad	4
VBROV-P-FAIRING-X1H	Carcasa protectora de espuma de flotabilidad	4
Number-4-75-selftappingscrew_90184A111	Tornillo	16
T200-THRUSTER-R1-CW	Turbinas	6
M3x16-socket-head-cap-screw-X1A_92290A120	Tornillo M3x16	21
BROV-M-BALLAST-200G-X1B	Lastre 200 g.	6
8-18-0625-self-tapping-screw-X1a_92525A820	Tornillo	6
PENETRATOR-M-NUT-10-LONG-R1_94036A625	Tornillo para conexionado de cables	3
PENETRATOR-M-BOLT-10-25-8MM-X1B_94036A625	Tornillo para conexionado de cables	3
PENETRATOR-M-BOLT-10-25-A-R2_94036A625	Tornillo para conexionado de cables	7
LUMEN-M-MOUNT-X1	Montura para foco	2
Assem-LUMEN-LIGHT-X2	Foco	2

Tabla 1. Lista de componentes del modelo 3D del BlueROV2.

El desarrollo de esta fase comienza exportando varias de las piezas principales, como son los paneles laterales y los cilindros de la electrónica y la batería.

En Autodesk se seleccionan los parámetros para la exportación de las piezas, como el refinamiento en valores medios para no sobrecargar la simulación en UWSim:

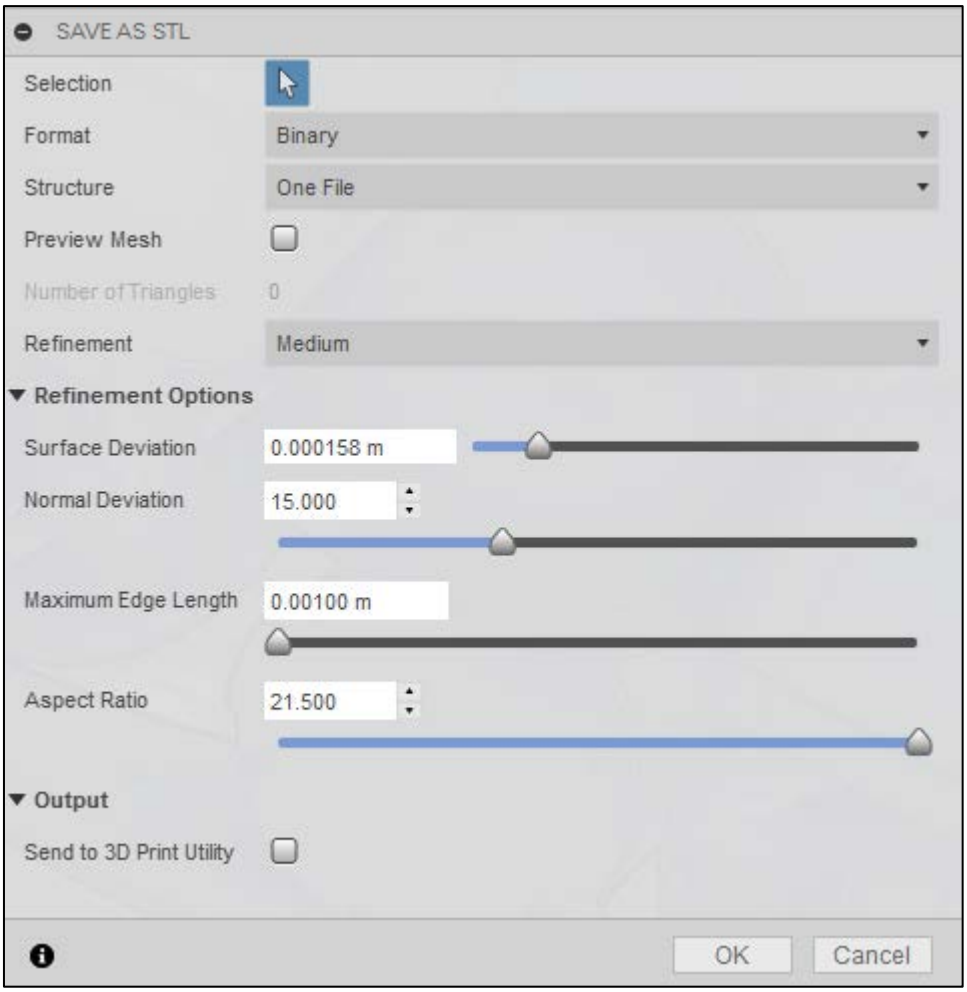


Figura 23. Parámetros para la exportación de las piezas en Autodesk Fusion 360.

Para que la simulación vaya lo más fluida posible (ya que durante la realización del trabajo se ha descubierto que el ordenador utilizado no es capaz de realizar simulaciones fluidas y con buen detalle con el modelo del ROV) se ha de simplificar al máximo los archivos, es decir, eliminar elementos que no sean meramente visuales. Esto aplica a tornillos, chips internos o tuercas insertables, entre otros.

La lista final simplificada de componentes que forman el conjunto del BlueROV2 es la siguiente:

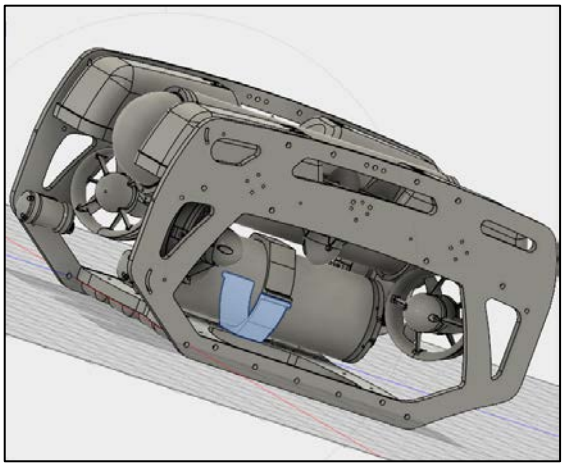


Figura 24. Soporte inferior del cilindro de la batería.

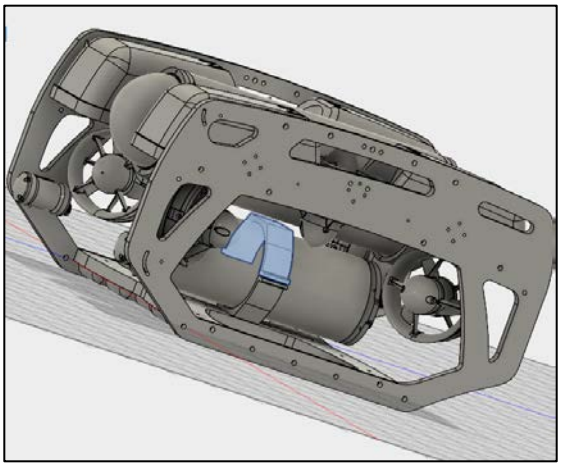


Figura 25. Soporte superior del cilindro de la batería.

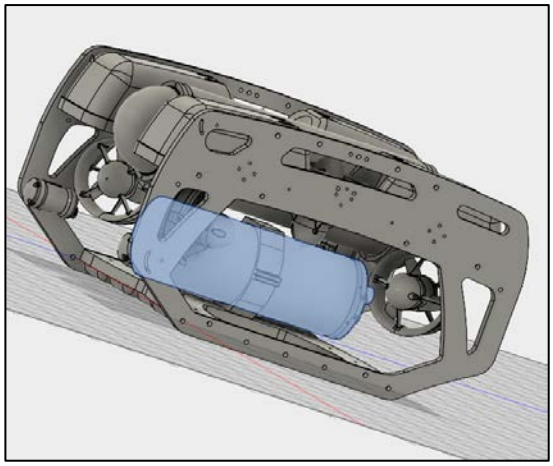


Figura 26. Cilindro inferior, que contiene la batería.

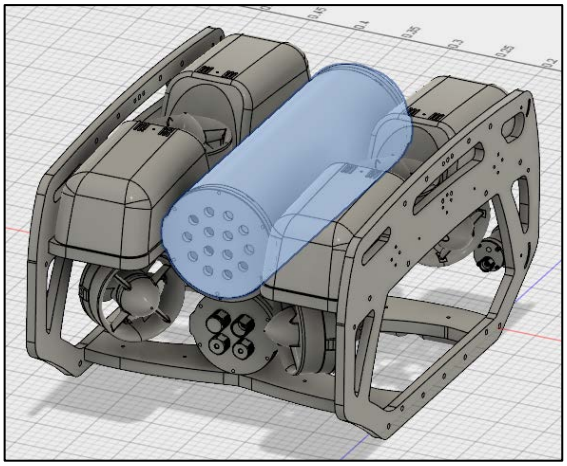


Figura 27. Cilindro superior, que contiene electrónica.

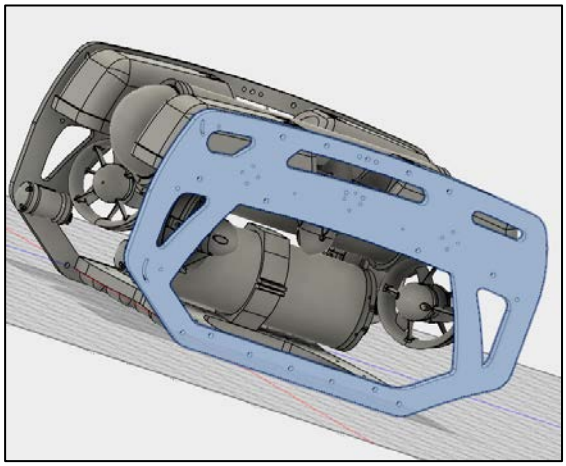


Figura 28. Panel lateral izquierdo.

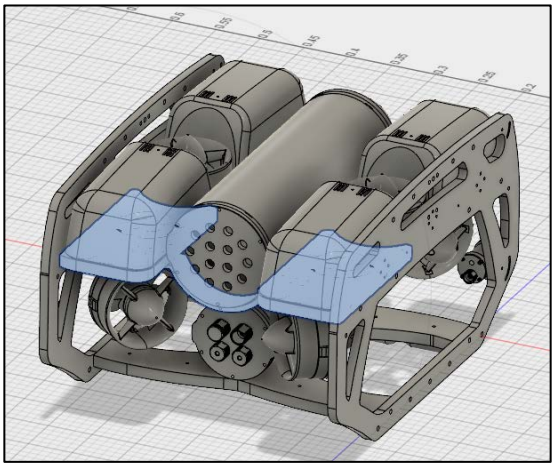


Figura 29. Panel horizontal trasero del chasis.

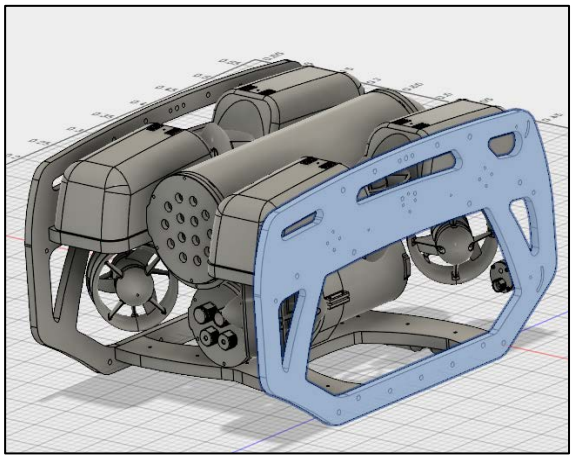


Figura 30. Panel lateral derecho.

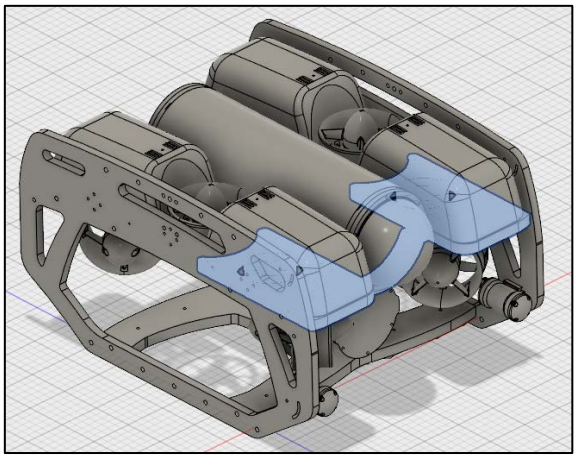


Figura 31. Panel horizontal delantero del chasis.

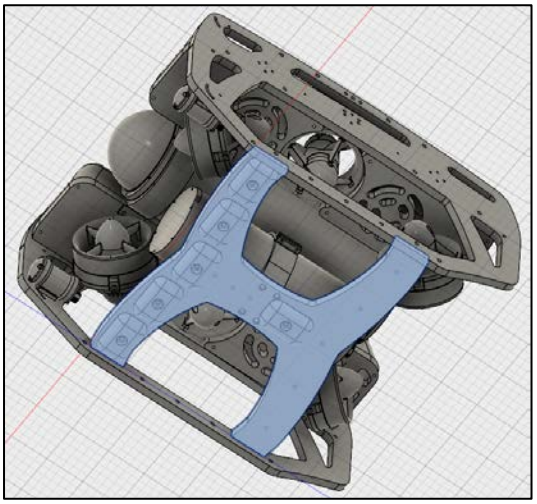


Figura 32. Panel inferior.

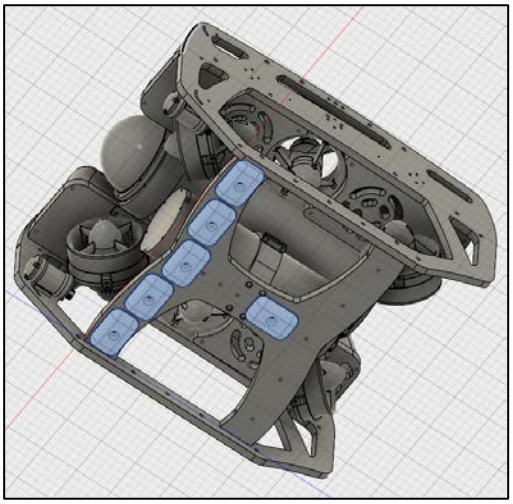


Figura 33. Lastres.

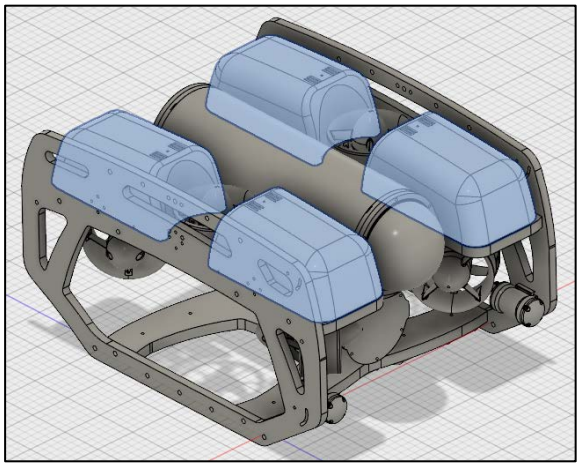


Figura 34. Carcasas protectoras de espuma de flotabilidad.

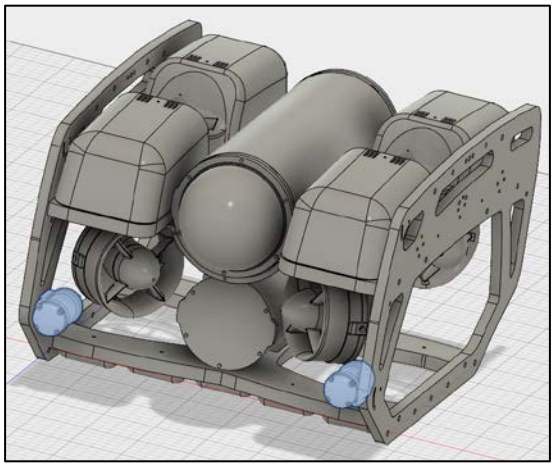


Figura 35. Focos de iluminación.

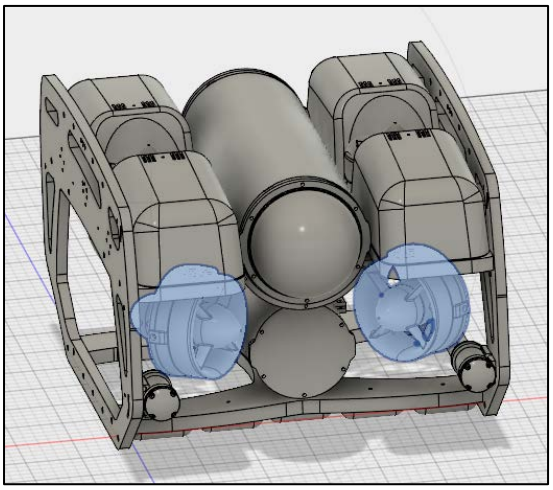


Figura 36. Turbinas delanteras.

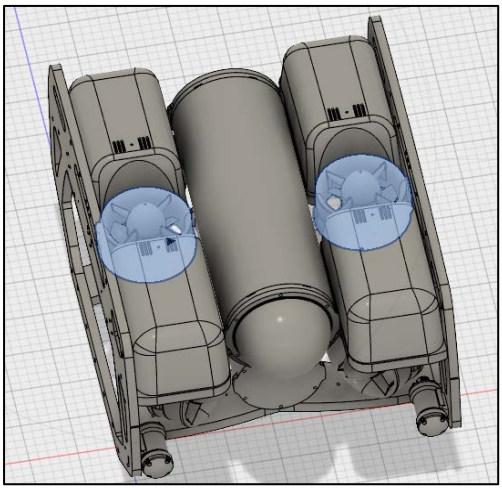


Figura 37. Turbinas superiores.

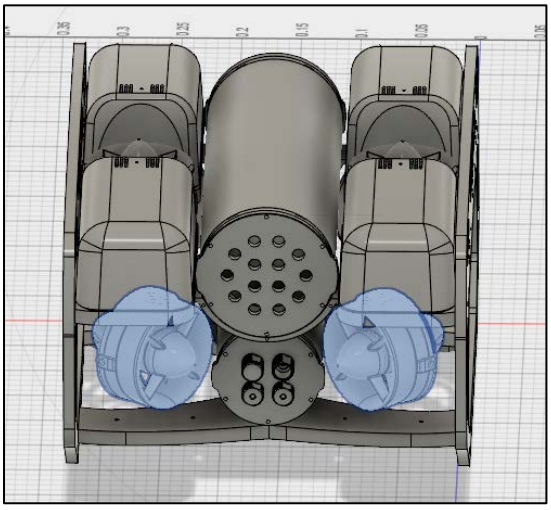


Figura 38. Turbinas traseras.

Finalmente, y a causa del problema encontrado para conformar el conjunto del robot sobre el UWSim pieza a pieza (hecho que se explica detalladamente en el siguiente capítulo), se decide exportar el conjunto simplificado del BlueROV2 en un solo archivo con extensión .stl

6.2. Archivo URDF

En este apartado se trata la realización del archivo que contiene la información del BlueROV2, bluerov2.urdf.

Un URDF (*Universal Robotic Description Format*) es un archivo XML usado en ROS para describir todos los elementos de los que se compone un robot (partes y articulaciones). En él se tienen que indicar posiciones y rotaciones, entre otros, de los componentes integrantes del vehículo.

Un archivo URDF sigue una estructura predefinida [40], y se explicará sobre un ejemplo con 2 piezas del BlueROV2:

```
1 <?xml version="1.0" ?>
2
3 <robot name="BlueROV2" xmlns:xacro="http://www.ros.org/wiki/xacro">
4   <link name="panel_inferior">
5     <inertial>
6       <mass value="xx"/>
7       <inertia ixx="xxx" ixy="xxx" ixz="xxx" iyy="xxx" iyz="xxx" izz="xxx"/>
8     </inertial>
9     <visual>
10      <origin rpy="0 0 0" xyz="0 0 0"/>
11      <geometry>
12        <mesh filename="robot/BLUEROV2/panel_inferior.stl"/>
13      </geometry>
14    </visual>
15  </link>
16
17  <link name="panel_lateral_derecho">
18    <inertial>
19      <mass value="xx"/>
20      <inertia ixx="xxx" ixy="xxx" ixz="xxx" iyy="xxx" iyz="xxx" izz="xxx"/>
21    </inertial>
22    <visual>
23      <origin rpy="x x x" xyz="x x x"/>
24      <geometry>
25        <mesh filename="robot/BLUEROV2/panel_lateral_derecho.stl"/>
26      </geometry>
27    </visual>
28  </link>
29
30  <joint name="panel_inferior_a_panel_lateral_derecho" type="fixed">
31    <parent link="panel_inferior"/>
32    <child link="panel_lateral_derecho"/>
33    <origin rpy="x x x" xyz="x x x"/>
34  </joint>
35
36 <robot>
```

Pieza 1

Pieza 2

Articulación 1-2

Figura 39. Ejemplo de URDF con 2 piezas del BlueROV2.

Los primeros dos bloques corresponden a las diferentes piezas, *links*, que conforman el robot. Se debe indicar tanto la masa, las inercias y origen de la pieza, como el objeto 3D que debe representar el simulador (*mesh)*.

El último bloque del ejemplo corresponde a la articulación (*joint*) que une las piezas 1 y 2. En todas las articulaciones se debe indicar siempre la pieza base, *parent link*, y la pieza a unir, *child link*, así como la posición y orientación de dicha articulación.

Debe indicarse también el tipo de articulación junto al nombre de la misma, pudiendo ser fija, continua o rotacional.

Este procedimiento es repetitivo para todas las piezas que conformen el ROV, indicando siempre orientaciones y posiciones tanto de las piezas como de las articulaciones.

Se encuentra una dificultad, y es la de colocar las piezas en una posición y orientación correctas unas respecto a otras para que el aspecto visual del vehículo en la simulación de UWSim sea aceptable. El problema radica en conformar la estructura de *joints* y *links* en un conjunto de 27 piezas como es este caso. En un inicio, pueden ir creándose las uniones, pero resulta imposible visualizar correctamente el resultado detallado en UWSim debido a la baja resolución y falta de fluidez del visor. Por ejemplo, un cambio de 10 cm en la posición de una pieza resulta casi imperceptible en el visor de UWSim, por lo que realmente no fue posible determinar cuándo una pieza estaba correctamente colocada en su lugar.

También resulta complicado establecer la posición de las *joints*, puesto que no se dispone de las medidas exactas de dichas uniones en el modelo real.

Por ello se valora la posibilidad de exportar el modelo entero en un solo archivo .stl.

Este hecho tiene la ventaja principal de que se consigue una representación del vehículo visualmente realista, con las piezas colocadas en su sitio exacto.

Por el contrario, se tiene una desventaja importante, y es que no se consigue un comportamiento dinámico personalizado para cada pieza, ya que en el planteamiento original se daría a cada pieza sus características dinámicas: momentos de inercia, centro de gravedad y masa.

Finalmente se decide optar por esta opción, creando el archivo .urdf compuesto por el bloque entero sin articulaciones, debido al deficiente resultado en el visor si se realiza el modelo pieza a pieza.

Como se detalló anteriormente, con Autodesk se realiza un cambio de materiales a algunas piezas tratando de conseguir una aproximación realista. Una vez realizado este paso, el conjunto del ROV tiene estas características:

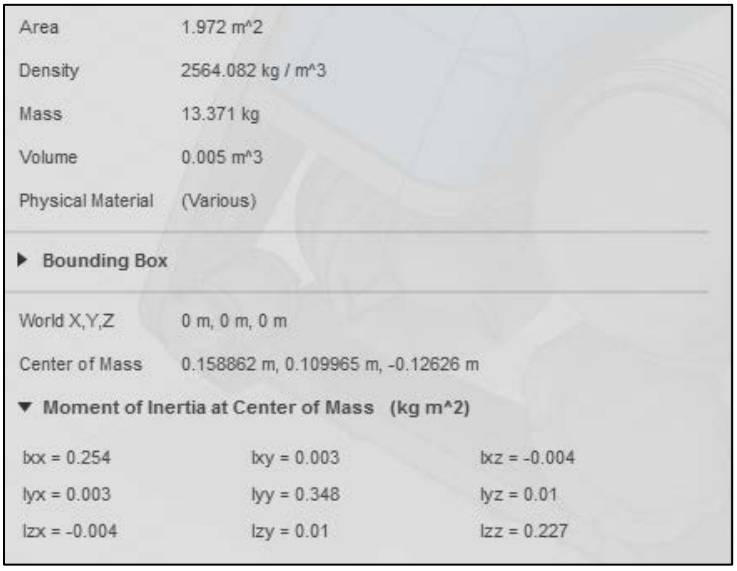


Figura 40. Características del ROV en Autodesk.

Por lo que el archivo final bluerov2.urdf contiene la siguiente información:

bluerov2.urdf × bluerov2scene.xml ×

```
<?xml version="1.0" ?>

<robot name="bluerov2" xmlns:xacro="http://www.ros.org/wiki/xacro">


  <link name="bluerov2full">
    <inertial>
      <mass value="13.371"/>
      <inertia ixx="0.254" ixy="0.003" ixz="-0.004" iyy="0.348" iyz="0.01" izz="0.227"/>
    </inertial>
    <visual>
      <origin rpy="-1.57 0 0" xyz="0 0 -2"/>
      <geometry>
        <mesh filename="robot/BLUEROV2/bluerov2full.stl"/>
      </geometry>
    </visual>
  </link>

</robot>

```

Figura 41. Bluerov2.urdf

6.3.Creación de la escena y la interfaz de comunicación de ROS

Como se explica anteriormente, la escena se trata de un archivo XML que contiene toda la información para la simulación como vehículos, texturas o sensores.

Dicha información está dividida en distintos bloques, entre los que destacan [41]:

- Bloque *oceanState*: permite configurar las características del agua en el simulador.
- Bloque *simParams*: permite configurar parámetros como sombras del vehículo, resolución del visor o las físicas de la simulación.
- Bloque de cámara: configura la cámara principal, es decir, la que visualiza la escena.
- El bloque de vehículos: configura tanto el vehículo como los sensores disponibles en él.
- El bloque de objetos: utilizado para añadir obstáculos u otros objetos a la escena, como barcos hundidos o cables marinos.
- El bloque *rosInterfaces*: permite configurar las comunicaciones de ROS con UWSim.

Existe la posibilidad de generar el archivo XML de la escena mediante archivos xacro, que permiten crear librerías predefinidas para sensores, vehículos u objetos.

Usando este método también se consigue un ahorro de tiempo y tamaño del archivo de la escena, pues simplemente se invocan a las librerías mediante el paso de parámetros para crear los distintos componentes, en vez de definir cada uno de ellos de manera independiente.

Dado que se iban a utilizar pocos elementos, se decide construir el archivo de la manera tradicional. No obstante, al final de este capítulo se comenta cómo se realizaría la escena en el caso de utilizar archivos xacro.

A continuación se detallan cada uno de ellos sobre el archivo *bluerov2scene.xml* creado para este trabajo.

bluerov2.urdf × bluerov2scene.xml ×

```
<?xml version="1.0"?>

<!DOCTYPE UWSimScene SYSTEM "UWSimScene.dtd" >

<UWSimScene>
  <oceanState>
    <windx> 0.04 </windx>
    <windy> 0.04 </windy>
    <windSpeed> 12 </windSpeed>
    <depth> 1000 </depth>
    <reflectionDamping> 0.35 </reflectionDamping>
    <waveScale> 1e-7 </waveScale>
    <isNotChoppy> 0 </isNotChoppy>
    <choppyFactor> 2.5 </choppyFactor>
    <crestFoamHeight> 2.2 </crestFoamHeight>
    <oceanSurfaceHeight> 0 </oceanSurfaceHeight>
    <fog>
      <density> 0.1</density>
      <color>
        <r>0</r>
        <g>0.05</g>
        <b>0.3</b>
      </color>
    </fog>
    <color>
      <r>0.0</r>
      <g>0.05</g>
      <b>0.3</b>
    </color>
    <attenuation>
      <r>0.015</r>
      <g>0.0075 </g>
      <b> 0.005 </b>
    </attenuation>
  </oceanState>
```

<oceanState>

En este bloque de código se configuran aspectos como:

- Velocidad y dirección del viento, que afecta a la altura de las olas.
- Color y visibilidad del agua con los *tags* <fog>, <color> y <attenuation>.

```
  <simParams>
    <disableShaders> 0 </disableShaders>
    <resw> 800 </resw>
    <resh> 600 </resh>
    <offsetp>
      <x>0</x>
      <y>0</y>
      <z>0</z>
    </offsetp>
    <offsetr>
      <x> 3.14</x>
      <y> 0</y>
      <z> -1.57 </z>
    </offsetr>
    <enablePhysics> 1 </enablePhysics>
    <showTrajectory>
      <target>bluerov2</target>
    </showTrajectory>
  </simParams>
```

<simParams>

Se configuran parámetros como las sombras, la resolución del visor (una menor resolución permite una mayor fluidez) o el eje de coordenadas (se configuran los parámetros de tal manera que el eje z positivo esté apuntando hacia el fondo del océano).

```
 <camera>
   <freeMotion> 1 </freeMotion>
   <objectToTrack>bluerov2/bluerov2full</objectToTrack>
   <fov> 60 </fov>
   <aspectRatio> 1.33 </aspectRatio>
   <near> 0.1 </near>
   <far> 10000 </far>
   <position>
     <x>-5</x>
     <y>-5 </y>
     <z>8 </z>
   </position>
   <lookAt>
     <x>0</x>
     <y>0 </y>
     <z>0 </z>
   </lookAt>
 </camera>
```

<camera>

Ajuste de la cámara que muestra la imagen del visor, seleccionando objeto a seguir, el campo de visión (fov) la posición y las distancias mínima y máxima de recorrido.

```
 73   <vehicle>
 74     <name>bluerov2</name>
 75     <file>data/scenes/bluerov2.urdf</file>
 76     <position>
 77       <x> 0</x>
 78       <y> 0 </y>
 79       <z> 0 </z>
 80     </position>
 81     <orientation>
 82       <r>0</r>
 83       <p>0</p>
 84       <y>0</y>
 85     </orientation>
```

<vehicle>

Se configura el ROV con distintos sub-bloques de código.
El nombre dado en el *tag* <name> es el que se usará para crear los *topics* de comunicación en ROS.

```
 86     <virtualCamera>
 87       <name>bowtech1</name>
 88       <relativeTo>bluerov2full</relativeTo>
 89       <resw> 320 </resw>
 90       <resh> 240 </resh>
 91       <position>
 92         <x> 0.1 </x>
 93         <y> 0.1 </y>
 94         <z> -0.2 </z>
 95       </position>
 96       <orientation>
 97         <r>0</r>
 98         <p>3.14</p>
 99         <y>1.57 </y>
100       </orientation>
101     </virtualCamera>
```

<virtualCamera>

Se añade una cámara simulada que apunta al fondo marino (ver Figura 44).

```
102     <rangeSensor>
103       <name>sonar</name>
104       <relativeTo>bluerov2full</relativeTo>
105       <range>10</range>
106       <visible>0</visible>
107       <position>
108         <x>0.1</x>
109         <y>0</y>
110         <z>0.2</z>
111       </position>
112       <orientation>
113         <r>0</r>
114         <p>-1.57</p>
115         <y>0</y>
116       </orientation>
117     </rangeSensor>
```

<rangeSensor>

Sensor de rango, con parámetros como la distancia de sensado, visibilidad del haz del sensor y la posición y orientación del origen del mismo.

```
118     <imu>
119       <name>imu</name>
120       <relativeTo>bluerov2full</relativeTo>
121       <position>
122         <x>0</x>
123         <y>0</y>
124         <z>0</z>
125       </position>
126       <orientation>
127         <r>0</r>
128         <p>0</p>
129         <y>0</y>
130       </orientation>
131       <std>0.00000001</std>
132     </imu>
```

<imu>

Estima la orientación del vehículo con respecto a los ejes de coordenadas.
<std> corresponde a la desviación estándar del ruido gaussiano añadido al dispositivo.

```
133     <pressureSensor>
134       <name>pressureSensor</name>
135       <relativeTo>bluerov2full</relativeTo>
136       <position>
137         <x>0.1</x>
138         <y>0.1</y>
139         <z>-0.2</z>
140       </position>
141       <orientation>
142         <r>0</r>
143         <p>0</p>
144         <y>0</y>
145       </orientation>
146       <std>0.02</std>
147     </pressureSensor>
```

<pressureSensor>

Sensor de presión. Se configuran tanto la posición/orientación como el objeto al que va ligado.

```
148     <gpsSensor>
149       <name>GPSSensor</name>
150       <relativeTo>bluerov2full</relativeTo>
151       <position>
152         <x>0</x>
153         <y>0</y>
154         <z>0</z>
155       </position>
156       <orientation>
157         <r>0</r>
158         <p>0</p>
159         <y>0</y>
160       </orientation>
161       <std>0.00005</std>
162     </gpsSensor>
```

<gpsSensor>

Dispositivo GPS. Se configuran tanto la posición/orientación como el objeto al que va ligado.

```
163     <dvlSensor>
164       <name>DVLSensor</name>
165       <relativeTo>bluerov2full</relativeTo>
166       <position>
167         <x>0</x>
168         <y>0</y>
169         <z>0</z>
170       </position>
171       <orientation>
172         <r>0</r>
173         <p>0</p>
174         <y>0</y>
175       </orientation>
176       <std>0.0015</std>
177     </dvlSensor>
```

<dvlSensor>

Sensor que estima la velocidad lineal del ROV.

```
178     <multibeamSensor>
179       <name>multibeam</name>
180       <relativeTo>bluerov2full</relativeTo>
181       <visible> 1 </visible>
182       <position>
183         <x> 0.1 </x>
184         <y> 0.1 </y>
185         <z> -0.2 </z>
186       </position>
187       <orientation>
188         <r> 0 </r>
189         <p> 1.57 </p>
190         <y> -1.57 </y>
191       </orientation>
192       <initAngle>-60</initAngle>
193       <finalAngle>60</finalAngle>
194       <angleIncr>0.1</angleIncr>
195       <range>0.5</range>
196     </multibeamSensor>
```

<multibeamSensor>

Haz de sensores. Se configuran parámetros como el emplazamiento, el recorrido angular que tiene, el número de haces que posee y su rango.
El *tag* <visible> se configura a 1 para que en la simulación se muestre este sensor.

```
197     <ForceSensor>
198       <name>ForceSensor</name>
199       <target>bluerov2full</target>
200       <offsetp>
201         <x> 0.1 </x>
202         <y> 0.1 </y>
203         <z> -0.2 </z>
204       </offsetp>
205       <offsetr>
206         <x>-1.57</x>
207         <y>0</y>
208         <z>3.14</z>
209       </offsetr>
210     </ForceSensor>
211   </vehicle>
```

<ForceSensor>

Sensor de fuerzas/inercias.
Se configuran la posición, orientación y el objeto al que va ligado.

```
234   <object>
235     <name>terrain</name>
236     <file>terrain/CIRS/cirs_trident.osg</file>
237     <position>
238       <x> 0</x>
239       <y> 0 </y>
240       <z> 0 </z>
241     </position>
242     <orientation>
243       <r>0</r>
244       <p>0</p>
245       <y>0</y>
246     </orientation>
247     <offsetp>
248       <x>-1.5</x>
249       <y>-3.0</y>
250       <z>0</z>
251     </offsetp>
252     <offsetr>
253       <x> 3.1415</x>
254       <y> 0</y>
255       <z> -1.57 </z>
256     </offsetr>
257     <physics>
258       <mass> 0 </mass>
259       <collisionShapeType> trimesh </collisionShapeType>
260     </physics>
261   </object>
```

<object>

En el archivo XML se pueden añadir diversos objetos 3D previamente modelados.
En este caso se opta por incluir una piscina de pruebas del IRSLab.

```
262   <object>
263     <name> blackbox </name>
264     <file> objects/blackbox_uib_trimesh.osg  </file>
265     <position>
266       <x> 0</x>
267       <y> 0 </y>
268       <z> 4.7 </z>
269     </position>
270     <orientation>
271       <r>0</r>
272       <p>3.1415</p>
273       <y>0</y>
274     </orientation>
275     <physics>
276       <mass> 15 </mass>
277       <inertia>
278         <x> 0</x>
279         <y> 0 </y>
280         <z> 0 </z>
281       </inertia>
282       <collisionShapeType> box </collisionShapeType>
283     </physics>
284   </object>
```

<object>

Este segundo objeto corresponde a una caja negra de un avión.

Con el vehículo correctamente configurado, se generan ahora los *topics*, o canales de traspaso de información, en el mismo archivo XML de la escena.

Cada sensor debe publicar la información que genera en un *topic* distinto, y después cada nodo interesado se debe suscribir a dicho *topic* para leer la información.

En este caso se ha optado por generar los siguientes:

```
<rosInterfaces>
  <ROSOdomToPAT>
    <topic> /dataNavigator </topic>
    <vehicleName> bluerov2 </vehicleName>
  </ROSOdomToPAT>
  <WorldToROSTF>
    <rootName> world </rootName>
    <enableObjects> 1 </enableObjects>
    <rate>10</rate>
  </WorldToROSTF>
  <VirtualCameraToROSImage>
    <cameraName>bowtech1</cameraName>
    <imageTopic> /uwsim/camera1 </imageTopic>
    <infoTopic> /uwsim/camera1_info </infoTopic>
  </VirtualCameraToROSImage>
  <VirtualCameraToROSImage>
    <cameraName>bowtech2</cameraName>
    <imageTopic> /uwsim/camera2 </imageTopic>
    <infoTopic> /uwsim/camera2_info </infoTopic>
  </VirtualCameraToROSImage>
  <ROSTwistToPAT>
    <topic> /BR2/twist</topic>
    <vehicleName> bluerov2 </vehicleName>
  </ROSTwistToPAT>
```

<ROSOdomToPAT>
Se suscribe al *topic* de odometría (dataNavigator) y aplica la posición o velocidad leída al vehículo.
<VirtualCameraToROSImage>
Publica la imagen de la cámara virtual y sus parámetros de calibración en dos *topics*.

```
  <RangeSensorToROSRange>
    <name>sonar</name>
    <topic> /uwsim/BR2/range </topic>
    <rate>10</rate>
  </RangeSensorToROSRange>
  <ROSPoseToPAT>
    <topic> /BR2/pose</topic>
    <vehicleName> bluerov2 </vehicleName>
  </ROSPoseToPAT>
  <ImuToROSImu>
    <name>imu</name>
    <topic>BR2/imu</topic>
    <rate>20</rate>
  </ImuToROSImu>
  <PressureSensorToROS>
    <name>pressureSensor</name>
    <topic>BR2/pressure</topic>
    <rate>5</rate>
  </PressureSensorToROS>
  <GPSSensorToROS>
    <name>GPSSensor</name>
    <topic>BR2/gps</topic>
    <rate>1</rate>
  </GPSSensorToROS>
```

<RangeSensorToROSRange>
Publica la información del sensor de rango.
<ROSPoseToPAT>
Se suscribe al *topic* de la posición del ROV y lo aplica a la simulación.
<ImuToROSImu>, <PressureSensorToROS> y <GPSSensorToROS>
Publican la información de sus sensores en los *topics*.

```
312     <DVLSensorToROS>
313       <name>DVLSensor</name>
314       <topic>BR2/dvl</topic>
315       <rate>5</rate>
316     </DVLSensorToROS>
317     <RangeImageSensorToROSImage>
318       <cameraName>rangeImage</cameraName>
319       <imageTopic> /uwsim/rangecamera </imageTopic>
320       <infoTopic> /uwsim/rangecamera_info </infoTopic>
321     </RangeImageSensorToROSImage>
322     <multibeamSensorToLaserScan>
323       <name>multibeam</name>
324       <topic>BR2/multibeam</topic>
325     </multibeamSensorToLaserScan>
326     <contactSensorToROS>
327       <name>bluerov2</name>
328       <topic>BR2/contactSensor</topic>
329       <rate> 100 </rate>
330     </contactSensorToROS>
331     <ForceSensorROS>
332       <name>ForceSensor</name>
333       <topic>BR2/ForceSensor</topic>
334       <rate>100</rate>
335     </ForceSensorROS>
336   </rosInterfaces>
337 </UWSimScene>
```

Las interfaces de la imagen de la izquierda realizan la misma función que los anteriormente mencionados, publicar la información recogida por los sensores en un *topic* independiente.

Como se comentó al inicio de este capítulo, si la escena se configura con múltiples objetos, vehículos y/o sensores, la opción de crear la escena mediante macros (archivos *xacro*) se torna casi obligada para ahorrar tiempo.

UWSim ofrece una orden para automáticamente crear los archivos .xml de las escenas a partir de archivos .xml.xacro presentes en la carpeta *scenes* utilizando el comando:

rosrun xacro xacro file.xacro > old.xml

Las macros más básicas que se utilizarían para crear la escena del presente trabajo son las siguientes:

- *Common*: macros genéricas.
- *DeviceLibrary:* con ella se generan sensores e interfaces de comunicación.
- *ObjectLibrary:* con ella se generan objetos como el terreno u obstáculos.
- *VehicleLibrary:* para crear vehículos. Esta librería hace uso de la librería *DevideLibrary* para acoplar los sensores al vehículo.

Con las librerías previas correctamente configuradas, únicamente haría falta configurar la macro que genera el archivo .xml de la escena, que para este trabajo se denominaría bluerov2scene.xml.xacro.

7. Resultados y conclusiones

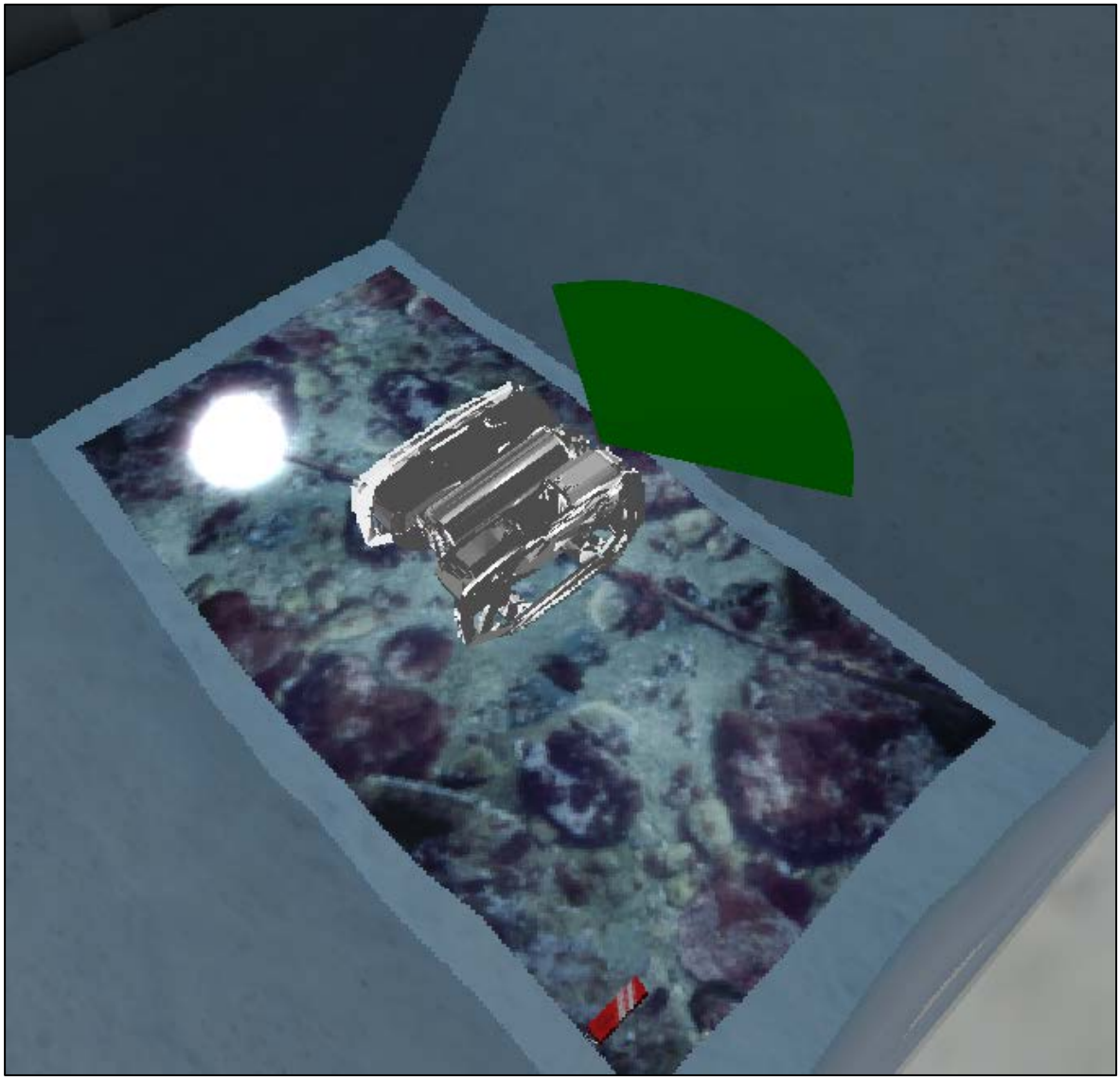


Figura 42. Resultado final, en el que se aprecia el ROV con el haz de sensores en el escenario con la caja negra sumergida.

En la figura superior se aprecia el resultado final obtenido en la escena. Se observa que la nitidez del modelo es baja, aunque se distingue la geometría básica. También se aprecia el haz de sensores del *mutlibeamSensor* en color verde, al tener el *tag* <visible> en 1, es decir, activado.

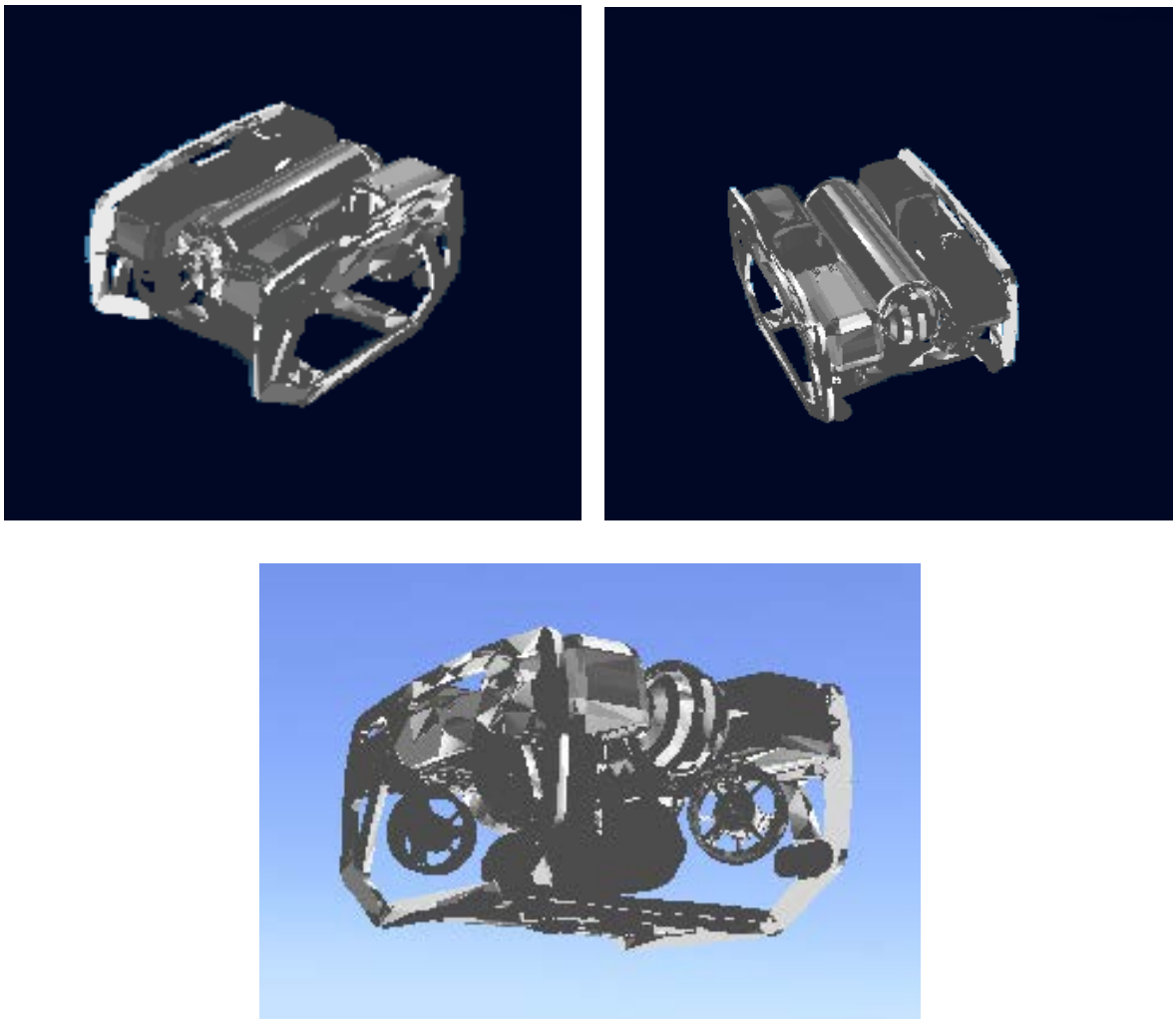


Figura 43. BlueROV2 en UWSim.

A pesar de que mediante imágenes es imposible demostrar, la fluidez de la simulación final es baja, por lo que se refuerza la decisión de haber exportado el ROV entero en un solo archivo, puesto que con el método de pieza a pieza se hubiese conseguido un peor resultado visual.

Posiblemente con un equipo con mayor capacidad de procesamiento se logre una simulación más fluida y un aspecto visual más realista, con un visor a mayor resolución que el aquí mostrado.

Para los *topics* creados se adjuntan capturas de la información que generan.

El primero de ellos corresponde a la cámara virtual (que se muestra pulsando la tecla “c”). Los nodos existentes, como el visor principal, se suscribe al *topic* necesario, en este caso el generado por la cámara 1, y muestra la imagen obtenida.

Figura 44. Imagen obtenida de la cámara simulada acoplada al ROV.

Estado del sensor de presión con el comando *rostopic echo /BR2/pressure*.

```
header:
  seq: 921
  stamp:
    secs: 1506100765
    nsecs: 392427039
  frame_id: world
pressure: -0.179372176528
---
header:
  seq: 922
  stamp:
    secs: 1506100765
    nsecs: 633708799
  frame_id: world
pressure: -0.203612089157
---
```

Estado del sensor de fuerza con el comando *rostopic echo /BR2/ForceSensor*.

```
header:
  seq: 26853
  stamp:
    secs: 1506100851
    nsecs: 720435548
  frame_id: bluerov2full
wrench:
  force:
    x: 0.0
    y: 0.0
    z: 0.0
  torque:
    x: 0.0
    y: 0.0
    z: 0.0
---
```

Estado del DVL con el comando *rostopic echo /BR2/dvl*

```
header:
  seq: 1781
  stamp:
    secs: 0
    nsecs:         0
  frame_id: ''
header_dvl: ''
date: ''
salinity: 0.0
temperature: 0.0
depth: 0.0
sound_speed: 0.0
test: 0
wi_x_axis: 0.0
wi_y_axis: 0.0
wi_z_axis: 0.0
wi_error: 0.0
wi_status: ''
bi_x_axis: 0.000124780025717
bi_y_axis: -0.000578003860256
bi_z_axis: -0.00124656060344
bi_error: 0.0
bi_status: ''
ws_transverse: 0.0
ws_longitudinal: 0.0
ws_normal: 0.0
ws_status: ''
bs_transverse: 0.0
bs_longitudinal: 0.0
bs_normal: 0.0
bs_status: ''
we_east: 0.0
we_north: 0.0
we_upwards: 0.0
we_status: ''
be_east: 0.0
be_north: 0.0
be_upwards: 0.0
be_status: ''
wd_east: 0.0
wd_north: 0.0
wd_upwards: 0.0
wd_range: 0.0
wd_time: 0.0
bd_east: 0.0
bd_north: 0.0
bd_upwards: 0.0
bd_range: 0.0
bd_time: 0.0
raw_data: ''
---
```

8. Futuros trabajos

Debido a los resultados finales obtenidos, y en pos de mejorar este trabajo, se plantean una serie de mejoras que incrementarían la precisión y fidelidad del modelo, tanto el aspecto visual como el comportamiento dinámico del mismo:

- Crear el modelo pieza a pieza. Se puede crear un modelo totalmente preciso en términos de movimiento y comportamiento dinámico obteniendo la masa real, momentos de inercia, centro de gravedad de cada pieza, y orientación y posición exactas respecto a la pieza colindante.
  El hecho de dotar a la espuma de flotabilidad o a los lastres de descenso de sus funciones reales dentro del simulador, entre otros muchos aspectos, permitiría un mayor realismo en las inmersiones virtuales.
- Unificar los Trabajos Fin de Grado paralelos realizados por terceros. Dado que el proyecto ha sido subdividido en varios trabajos, una futura tarea imprescindible es la de unificar dichos desarrollos. Con esto se conseguirían simulaciones completas sobre trayectorias establecidas, incluyendo el sistema en bucle cerrado de reacción ante obstáculos.
  Dicha integración tendría varios puntos vitales:
  - La información recogida por los sensores de este trabajo como cámaras, sensores de barrido o sónar, entre otros, se emplearía en el sistema de reacción ante objetos.
  - La información proveniente de la planificación de trayectorias sería leída e interpretada para crear el movimiento de los 6 propulsores disponibles.
  - Toda esta información en bucle cerrado se transmitiría y recibiría mediante *topics* en ROS; unificando dichos *topics* se puede agrupar todo el trabajo dividido realizado y conseguir que el modelo virtualizado se comporte con la mayor precisión posible respecto a como lo haría un modelo real en un entorno submarino.

9. Entorno socio-económico

9.1. Impacto socio-económico

A continuación se realiza un análisis del impacto que tendría la implantación satisfactoria de este proyecto en la vida real. También se tienen en cuenta los impactos generados por el uso de este tipo de vehículos en otras operaciones distintas al objetivo del proyecto STAMS, como la investigación de flora y fauna marítima o mantenimiento de cables submarinos.

A grandes rasgos, la mayor ventaja de utilizar este tipo de vehículos es la posibilidad y facilidad de realizar operaciones a gran profundidad antes impensables, o la no necesidad de emplear buzos para las mismas.

Como impactos positivos se encuentran:

- Monitorización y mapeado de minas inundadas, realizando un estudio que permita predecir y evitar catástrofes como el colapso de una mina que tenga población civil sobre la superficie de la misma.
- Investigación de flora y fauna marinas. Mediante su estudio se puede evitar en cierta medida la destrucción del entorno marino en proyectos como la colocación de cables submarinos, estaciones de generación de energía mareomotriz o eólica e implantación de plataformas de extracción de petróleo.
- Estudio de hundimientos y recuperación de objetos. Con este tipo de vehículos se puede estudiar el estado de navíos hundidos, comprobar el estado de los depósitos de combustible o tanques de sustancias tóxicas y así poder valorar correctamente una operación con buzos sin grandes riesgos.
- Facilidad y rapidez en la obtención de la información. Al tener la posibilidad de recibir la información recogida por sensores y cámaras en tiempo real, se pueden realizar decisiones in situ con un equipo de expertos, sin esperar a que el buzo emerja y detalle la información.
- Ahorro económico con las simulaciones. Con unos simuladores realistas se prepara al piloto del vehículo a manejar situaciones habituales, como la operación con brazos robotizados, y también imprevistos, como el enredo

del cable con objetos del fondo marino que puedan inutilizar el ROV y posibilitar su pérdida.
- Eliminación de peligros para los buzos, pues su inmersión ya no es necesaria en muchas de las operaciones.

Como impactos negativos se encuentran:

- Roturas de tuberías o cables submarinos por imprecisiones en el uso de los vehículos, o imprevistos relacionados con corrientes de agua o malfuncionamientos. Estos casos pueden derivar en la salida de materiales tóxicos al ambiente, provocando contaminación del entorno.
- Posibilidad de pérdida del vehículo por malfuncionamiento o error del operario, lo que generaría contaminación del entorno, puesto que los vehículos llevan componentes electrónicos y materiales tóxicos como plásticos o baterías.
- Alteración del ecosistema del entorno de funcionamiento, ya que los propulsores generan contaminación acústica que puede afectar al comportamiento de la fauna marina existente en la zona.

9.2. Presupuesto

El presupuesto para este trabajo se divide en 2 partes, una de material utilizado para su realización y otra con el personal que es necesario para llevarlo a cabo.

La primera incluye 3 componentes *open-source*, esto es, de código abierto y sin coste de utilización, que junto a la licencia gratuita para la comunidad educativa de Autodesk Fusion 360, hacen que el presupuesto destinado a equipo informático sea reducido.

Para el ordenador empleado sólo se tiene en cuenta el coste de amortización. Su precio de compra es de 599€, y se estima una vida útil de 6 años, y un valor residual de 0€, lo que se traduce en un coste de 100€/año.

Se toma como duración del proyecto un año, para realizar pruebas suficientes y análisis.

ÍTEM	DESCRIPCIÓN	CANTIDAD	PRECIO [€]
1	Ordenador portátil HP Pavilion G6 (amortización)	1	100€
2	Memoria flash USB 3.0 SanDisk Ultra Fit 32 GB	1	13,95
3	Licencia educativa Autodesk Fusion 360	1	0 €
4	Sistema Operativo Ubuntu 14.04.5 LTS	1	0 €
5	ROS Indigo	1	0 €
6	UWSim 1.4	1	0 €
	PRESUPUESTO DE MATERIAL INFORMÁTICO		113,95 €

Tabla 2. Presupuesto para material informático.

A continuación se detalla el presupuesto destinado para personal necesario para la elaboración del presente trabajo, desde la fase de estudio previo hasta la implementación y documentación:

TAREA	RESPONSABLE	Coste [€]/h	Horas	PRECIO [€]
Estudio previo	Ingeniero Técnico	20	50	1000 €
Concepción e implementación	Ingeniero Técnico	20	100	2000 €
Documentación	Administrativo	12,5	125	1562,5 €
	PRESUPUESTO DE PERSONAL			4562,5 €

Tabla 3. Presupuesto para personal.

Anexos

Anexo I: Planificación

Este trabajo empezó a realizarse en el mes de abril de 2017. Los primeros pasos fueron de investigación, tanto del objetivo del proyecto como del estado de proyectos similares, y de aprendizaje de las herramientas que se utilizan.

De dicho período inicial, el aprendizaje de ROS y UWSim abarcó gran parte del tiempo, debido al desconocimiento en el sistema operativo Linux y ROS, resultando compleja la adaptación al sistema de comandos en terminal de Linux.

Posteriormente, tras la instalación del software necesario en ambos sistemas operativos, anteriormente mencionado, se comenzó a la implantación del modelo del BlueROV2 en UWSim.

Este paso tuvo numerosas dificultades ya mencionadas en páginas previas, por lo que llevó más tiempo del inicialmente contemplado.

El diseño de la escena también llevó un tiempo prolongado debido al desconocimiento de programación en lenguaje XML y continua investigación para los sensores y *topics* creados.

Por último, la redacción de este documento abarcó un tiempo notable, pues se ha llevado a cabo simultáneamente con otras tareas para no ir perdiendo detalles del proceso.

Se adjunta un diagrama de Gantt que ilustra los procesos y períodos abarcados.

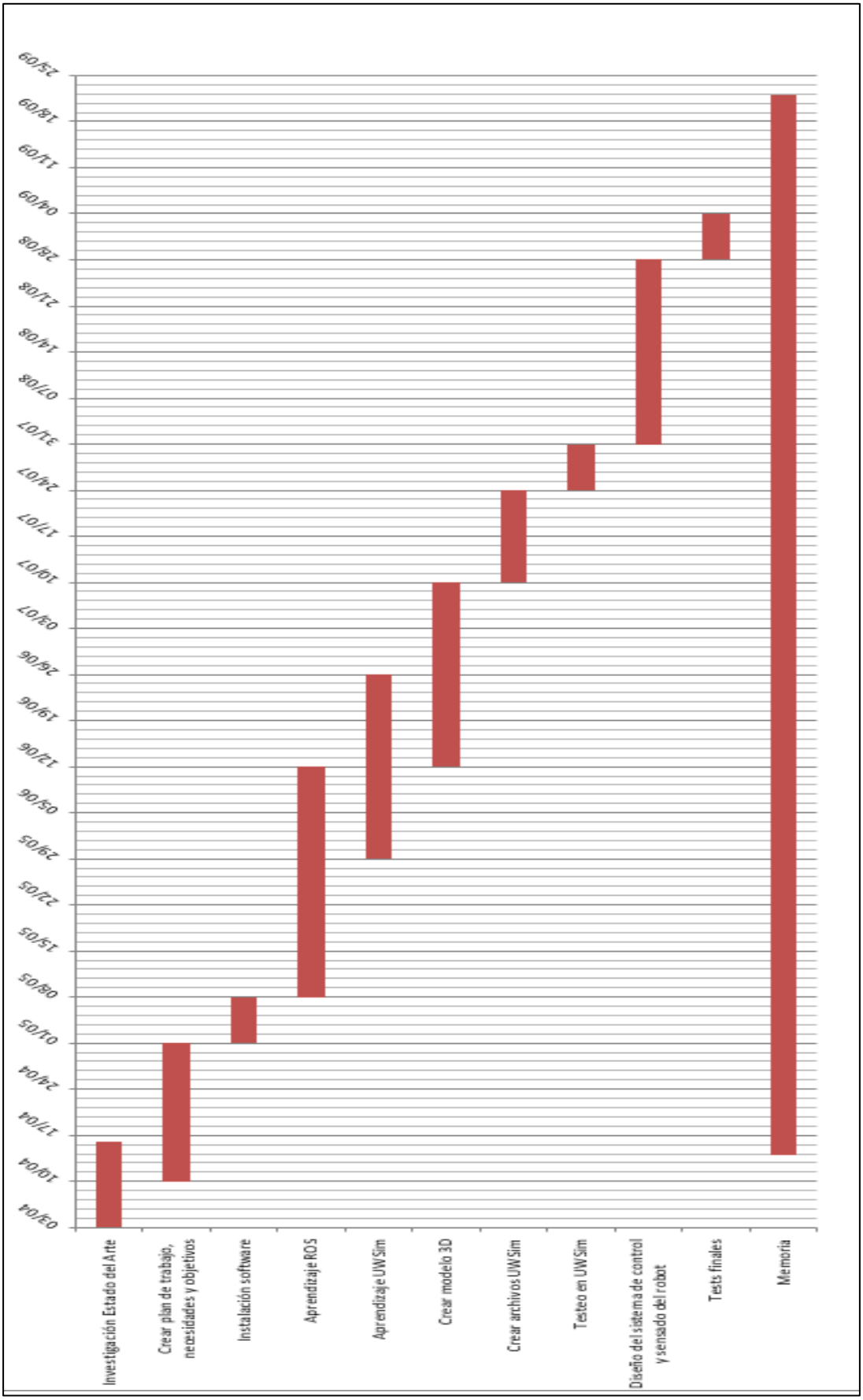


Figura 45. Diagrama de Gantt.

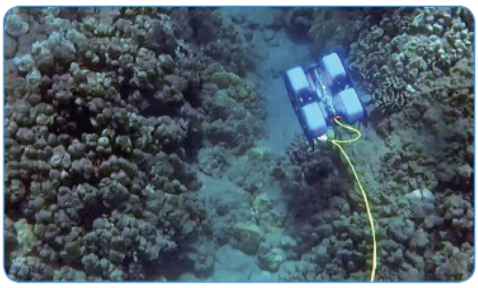
Anexo II: Hoja de características del BlueROV2

Datasheet

Blue Robotics

BlueROV2 Technical Specifications

Revision 10/16



Physical

Length	457 mm	18 in
Width	338 mm	13.3 in
Height	254 mm	10 in
Weight in Air *(with Ballast)*	10-11 kg	22-24 lb
Weight in Air *(without Ballast)*	9-10 kg	20-22 lb
Net Buoyancy *(with Ballast)*	0.2 kg	0.5 lb
Net Buoyancy *(without Ballast)*	1.4 kg	3 lb
Watertight Enclosure Inner Diameter	102 mm	4 in
Watertight Enclosure Inner Length	298 mm	11.75 in
Cable Penetrator Holes	14 x 10 mm	14 x 0.4 in
Construction	HDPE frame, aluminum flanges/end cap, & acrylic tubes	
Main Tube *(Electronics Enclosure)*	Blue Robotics 4" Series w/ aluminum end caps	
Battery Tube	Blue Robotics 3" Series w/ aluminum end caps	
Buoyancy Foam	R-3318 urethane foam rated to 210 m	
Ballast Weight	6 x 200 g coated lead weights	
Battery Connector	XT90	

Performance

Maximum Rated Depth	100 m	330 ft
Maximum Tested Depth *(so far)*	130 m	425 ft
Maximum Forward Speed	1 m/s	2 knots
Thrusters	Blue Robotics T200	
ESC	Blue Robotics Basic 30A ESC	
Thruster Configuration	6 thrusters	
	- 4 Vectored	
	- 2 Vertical	
Forward Bollard Thrust	14 kgf	30 lbf
Vertical Bollard Thrust	9 kgf	20 lbf
Lateral Bollard Thrust	14 kgf	30 lbf

Tether

Diameter	7.6 mm	0.30 in
Length	25-300 m	80-980 ft
Working Strength	45 kgf	100 lbf
Breaking Strength	160 kgf	350 lbf
Strength Member	Kevlar with waterblock	
Buoyancy in Freshwater	Neutral	
Buoyancy in Saltwater	Slightly positive	
Conductors	4 twisted pairs, 26 AWG	

Lights

Brightness	2 or 4 x 1500 lumens each with dimming control
Light Beam Angle	135 degrees, with adjustable tilt

Camera

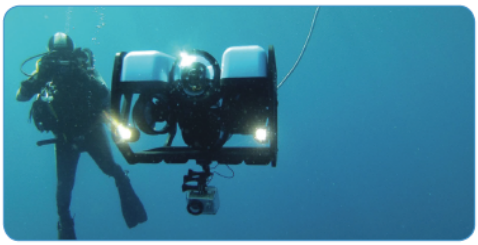
Camera	1080p digital
Camera Field of View	110 degrees horizontally
Tilt Range	+/- 90 degree camera tilt *(180 total range)*
Tilt Servo	Hitec HS-5055MG

Sensors

- 3-DOF Gyroscope
- 3-DOF Accelerometer
- 3-DOF Magnetometer
- Internal barometer
- Blue Robotics Bar 30 Pressure/Depth & Temperature Sensor *(external)*
- Current and Voltage Sensing
- Leak Detection

Battery *(can be changed in about 30 seconds)*

Battery Life *(Normal Use)*	2-3 hours w/ 18Ah battery
Battery Life *(Light Use)*	4-6 hours w/ 18Ah battery



BlueRobotics 4030 Spencer Street, Suite 102 // Torrance, CA 90503 // info@bluerobotics.com bluerobotics.com

Figura 46. Hoja de características del BlueROV2 [38]

Anexo III: Instalación de Ubuntu 14.04.5 LTS

Debido a la falta de este Sistema Operativo en el ordenador empleado para el trabajo, se realiza la instalación del mismo en las primeras fases.

Se instala junto al Sistema Operativo Windows 10 Pro, en el que se realizan otras tareas del trabajo.

El primer paso es descargar la imagen Ubuntu 14.04 del servidor [42] [43], y mediante la herramienta Rufus 2.3.709 crear una memoria flash booteable con el SO en su interior para su instalación.

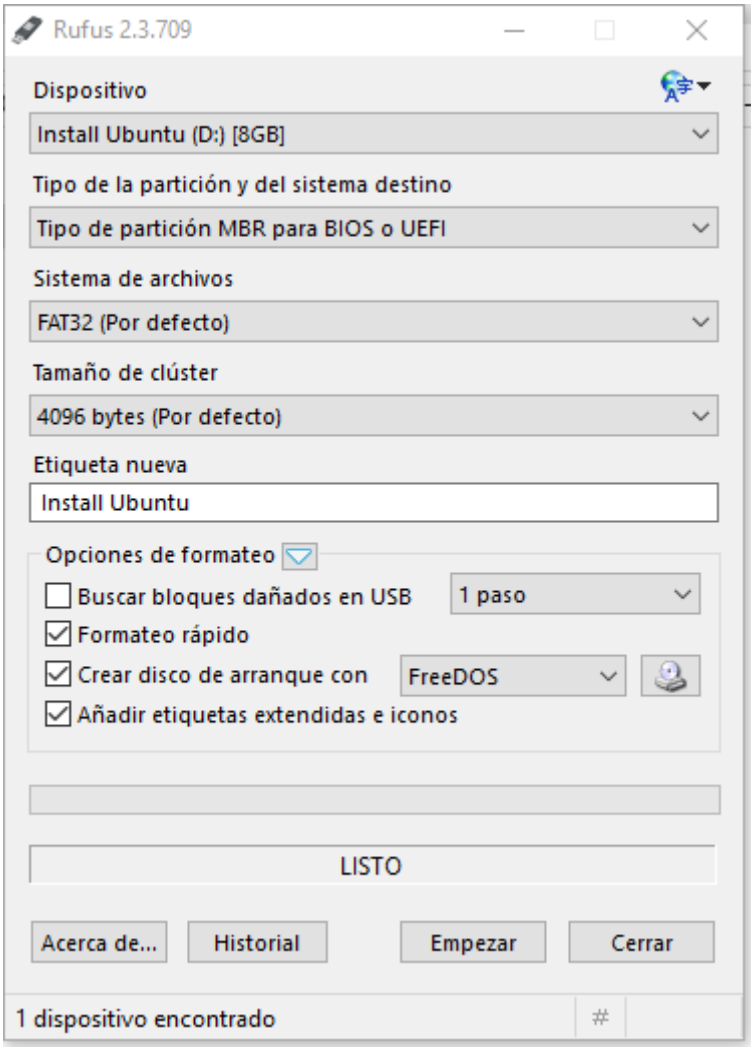


Figura 47. Creación del dispositivo booteable con Ubuntu 14.04.5 LTS

A continuación, se enciende el ordenador entrando en la BIOS, donde se escoge la opción de iniciar desde el dispositivo flash utilizado en el anterior paso.

Cabe resaltar que no es necesario hacer una partición del disco duro previamente en Windows 10, puesto que el instalador de Ubuntu 14.04.5 LTS dispone de dicha opción.

Tras elegir el idioma deseado (siendo el inglés el recomendado, debido a que la mayoría de tutoriales y ayuda en internet están disponibles en dicho idioma), se ha de seleccionar la opción “Instalar Ubuntu junto a Windows 10”, para que la instalación respete los archivos y configuraciones del otro Sistema Operativo presente en el disco duro.

Figura 48. Instalación del sistema Operativo Ubuntu 14.04.5 LTS

Tras la asignación del espacio deseado, se configuran los parámetros requeridos, como el nombre del equipo, contraseña o idioma del teclado y se finaliza con la instalación del Ubuntu 14.04.5 LTS.

Anexo IV: Instalación de ROS Indigo

Su instalación se realiza por la terminal de comandos de Ubuntu, siguiendo los pasos determinados en su web [44].

El paso cero, o paso previo a la instalación, es habilitar los 4 principales repositorios de software de Ubuntu. Un repositorio es un archivo que contiene diferente software de Ubuntu que puede ser instalado. Hay 2 tipos principales de repositorios: offline y online [45].

Con la instalación de Ubuntu 14.04.5 LTS se habilitan automáticamente los repositorios offline MAIN y RESTRICTED.

Para la instalación de Ros Indigo se deben habilitar además 2 repositorios online: UNIVERSE y MULTIVERSE.

Esta tarea puede hacerse de dos maneras:

1)En los ajustes del sistema: accediendo a los ajustes de sistema, *System Settings*, y en el apartado *System* se selecciona la opción *Software & Updates.*

Se han de habilitar los 4 repositorios principales.

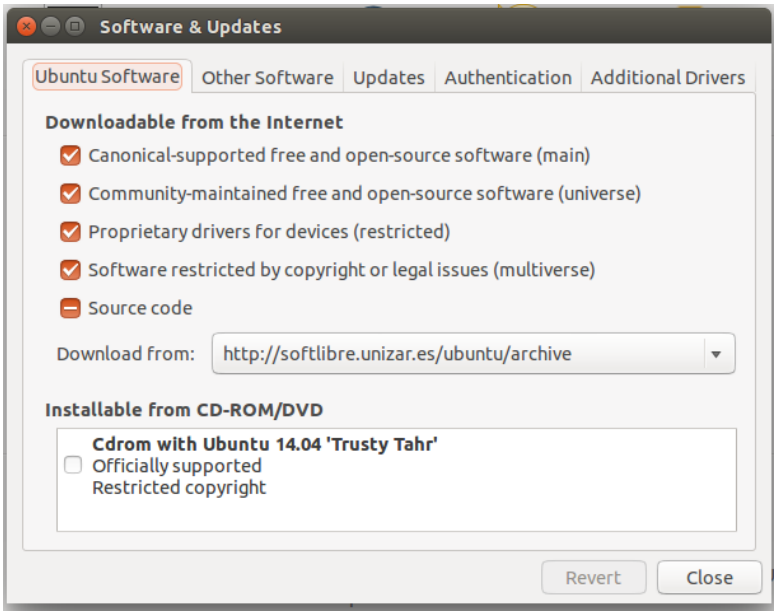


Figura 49. Repositorios de Ubuntu

2) Por terminal. Se habilitarán los repositorios en modo superusuario (para ello basta con añadir *sudo* antes de cada comando) mediante los siguientes comandos:

sudo add-apt-repository main

sudo add-apt-repository restricted

sudo add-apt-repository universe

sudo add-apt-repository multiverse

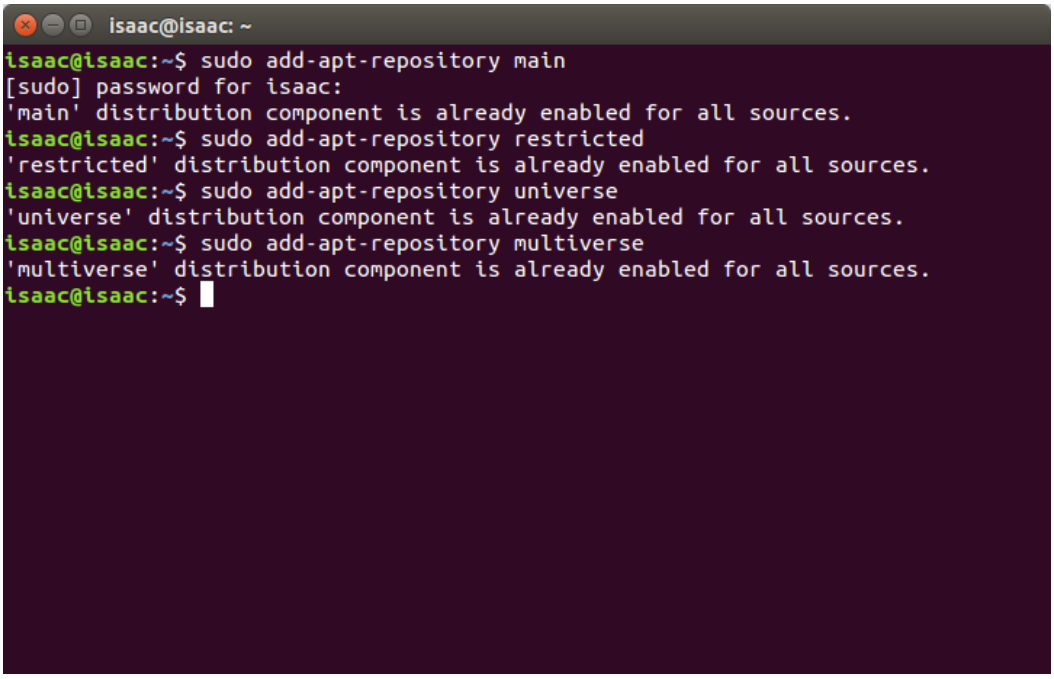


Figura 50. Habilitar los repositorios mediante comandos en la terminal

Se actualizan los paquetes de Debian:

```
isaac@isaac:~$ sudo apt-get update
```

Se instala la versión completa de ROS Indigo, que instala además de ROS, RVIZ y algunas librerías adicionales:

```
isaac@isaac:~$ sudo apt-get install ros-indigo-desktop-full
```

Se actualizan los paquetes de Debian de nuevo.

A continuación, se inicia y actualiza *rosdep*, que facilita la instalación de las dependencias necesarias para trabajar con ROS:

```
isaac@isaac:~$ sudo rosdep init
```

```
isaac@isaac:~$ rosdep update
```

Posteriormente, se debe configurar convenientemente la sesión *bash* que se esté usando (siendo *bash* el software que interpreta las órdenes de comandos de la terminal de Ubuntu). Para ello se lanzan los siguientes comandos:

```
isaac@isaac:~$ echo "source /opt/ros/indigo/setup.bash" >> ~/.bashrc
```

```
isaac@isaac:~$ source  ~/.bashrc
```

Es recomendable que cada vez que se inicie ROS se configure correctamente el *bash* mediante el comando:

```
isaac@isaac:~$ source /opt/ros/indigo/setup.bash
```

Se actualizan los paquetes de Debian de nuevo.

Por último, es recomendable instalar una herramienta de ROS llamada *rosinstall*, que permite instalar de manera sencilla paquetes de ROS adicionales a través de la terminal:

```
isaac@isaac:~$ sudo apt-get install phyton-rosinstall
```

Anexo V: Instalación de UWSim

Su instalación se realiza mediante la terminal de Ubuntu, siguiendo los pasos detallados por los propios desarrolladores [46].

De las diferentes opciones de instalación, se opta por instalar la "FULL SOURCE-BASED INSTALLATION", ya que contiene todas las librerías disponibles y en la última versión estable lanzada por los desarrolladores.

En primer lugar, se crea un archivo vacío con extensión .rosinstall en el espacio de trabajo *catkin_ws* (ver Anexo VI) que se vaya a utilizar, y se añade el siguiente contenido en su interior:

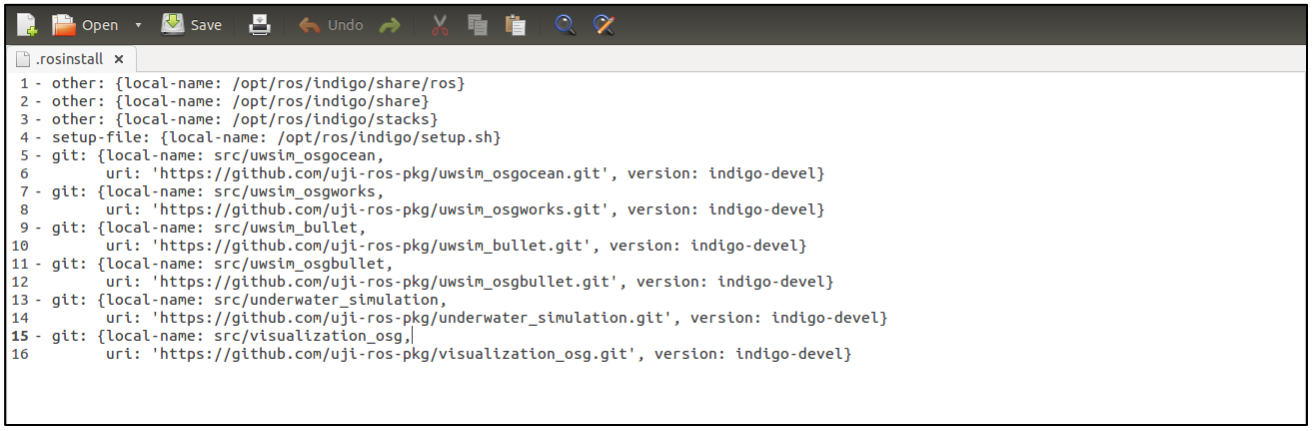


Figura 51. Contenido del archivo de instalación .rosinstall

Posteriormente se han de ejecutar los siguientes comandos en la terminal:

```
rosws update
```

Después el siguiente comando para instalar las dependencias con catkin:

```
rosdep install --from-paths src --ignore-src --rosdistro groovy -y
```

Y, por último:

```
catkin_make_isolated --install
```

Anexo VI: Creación del espacio de trabajo catkin_ws

Catkin es el compilador oficial de ROS, instaurado como sucesor del compilador original, rosbuild.

Viene instalado por defecto cuando se instala una distribución de ROS, por lo que no es necesario detallar una instalación para este concepto en particular.

Catkin combina macros originarias de CMake y *scripts* de Python para añadir algunas funcionalidades extras al sistema de trabajo de CMake, como el soporte para la búsqueda automática de paquetes de software o compilaciones simultáneas de múltiples proyectos dependientes.

El primer comando necesario es el propio de la creación del espacio de trabajo catkin:

```
mkdir -p ~/catkin_ws/src
```

Posteriormente se ha de ejecutar:

```
cd ~/catkin_ws/
```

```
catkin_make
```

catkin_make es una herramienta para trabajar con espacios de trabajo catkin. La primera vez que se ejecuta crea un archivo CMakeLists.txt en la carpeta *src* del espacio de trabajo, además de crear las carpetas *build* y *devel*, esta última con archivos de configuración setup.*sh.

El último comando necesario para comenzar a trabajar en el espacio de trabajo es:

```
source devel/setup.bash
```

Bibliografía

[1] ROV.org - Evolución de los ROV. (Última vez visitado: 4 de mayo de 2017): http://www.rov.org/rov_history.cfm

[2] Google Imágenes – "CURV III" (Última vez visitado: 5 de mayo de 2017): https://fas.org/irp/program/collect/curv3.jpg

[3] Google Imágenes – "RCV-225" (Última vez visitado: 5 de mayo de 2017): http://www.subaquaticamagazine.es/wp-content/uploads/2017/08/rcv-225-con-brazo-articulador.jpg

[4] Google Imágenes – "ROV Kaiko" (Última vez visitado: 5 de mayo de 2017): https://3.bp.blogspot.com/-auU4ruogmuc/VejEH7Wb4TI/AAAAAAAABNY/fJkydc9Eonc/s1600/Kaiko7000II.jpg

[5] Rovexchange.com - Descripción y categorías de los ROV. (Última vez visitado: 4 de mayo de 2017): http://www.rovexchange.com/mc_rov_overview.php

[6] ROV.org - Categorías de ROV. (Última vez visitado: 4 de mayo de 2017): http://www.rov.org/rov_categories.cfm

[7] Google Imágenes – "Quest LCROV" (Última vez visitado: 5 de mayo de 2017): http://www.rov.org/rov_applications_images/quest_lcrov.jpg

[8] Google Imágenes – "ROV Viper" (Última vez visitado: 5 de mayo de 2017): http://www.rov.net/pages/Viper.jpg

[9] Google Imágenes – "ROV Triton XL" (Última vez visitado: 5 de mayo de 2017): http://www.rov.net/images/Triton_XL.jpg

[10] Google Imágenes – "Cable plough" (Última vez visitado: 5 de mayo de 2017): https://i.imgur.com/HAcEyYF.jpg

[11] ROV.org – Consideraciones de diseño de un ROV. (Última vez visitado: 13 de mayo de 2017): http://www.rov.org/rov_design_overview.cfm

[12] Google Imágenes – "ROV design spiral" (Última vez visitado: 14 de mayo de 2017): http://www.rov.org/rov_applications_images/design_overview_chart.gif

[13] Google Imágenes – "Relative illumination vs distance" (Última vez visitado: 14 de mayo de 2017): www.rov.org/rov_applications_images/lighting_illumination_chart_tn.jpg

[14] ROV.org – Aplicaciones actuales de los ROV. (Última vez visitado: 13 de mayo de 2017): http://www.rov.org/rov_applications.cfm

[15] Elmundo.es – "El cable de fibra que unirá Bilbao y EEUU comenzará a desplegarse el 12 de junio" (Última vez visitado: 8 de junio de 2017): http://www.elmundo.es/economia/innovadores/2017/05/31/592e5914268e3ef5238b4576.html

[16] ROV.org – Aplicaciones científicas y académicas de los ROV. (Última vez visitado: 13 de mayo de 2017): http://www.rov.org/rov_academic.cfm

[17] ROV.org – Aplicaciones militares de los ROV. (Última vez visitado: 13 de mayo de 2017): http://www.rov.org/rov_military.cfm

[18] ROV.org – Zonas actuales de operación con ROV. (Última vez visitado: 13 de mayo de 2017): http://www.rov.org/rov_locations.cfm

[19] Sputniknews.com – "Rusia extraerá en 2030 más de la mitad del petróleo y del gas árticos". (Última vez visitado: 15 de julio de 2017): https://mundo.sputniknews.com/prensa/201507031038930138/

[20] Gazebosim.org – Plugins para simular comportamientos subacuáticos en objetos. (Última vez visitado: 21 de mayo de 2017): http://gazebosim.org/tutorials?tut=hydrodynamics&cat=plugins

[21] Marinesimulation.com – Simulador submarino ROVSim² PRO. (Última vez visitado: 21 de mayo de 2017): http://marinesimulation.com/rovsim%c2%b2-pro/

[22] Google Imágenes – "ROVSim2 PRO" (Última vez visitado: 21 de mayo de 2017): http://marinesimulation.com/marsimwp/wp-content/uploads/2013/06/ROVsim2_Pro_912.jpg

[23] ROVS.es – Simulador submarino VMAX ROV Simulator. (Última vez visitado: 21 de mayo de 2017): http://www.rovs.es/vmax-rov-simulator

[24] Google Imágenes – "VMAX ROV" (Última vez visitado: 21 de mayo de 2017): https://static.wixstatic.com/media/812f0e_96a52333e293d6fc8e813e6b4c31248e.png/v1/fill/w_345,h_260,al_c,usm_0.66_1.00_0.01/812f0e_96a52333e293d6fc8e813e6b4c31248e.png

[25] Arisimulation.com – Simulador submarino ARI ROV para operaciones de extracción de petróleo y gas. (Última vez visitado: 21 de mayo de 2017): http://www.arisimulation.com/products/oil-and-gas-offshore-simulators/rov-operations

[26] Google Imágenes – "ARI Simulation ROV" (Última vez visitado: 21 de mayo de 2017): http://media-s3.viva-images.com/vivastreet_in/clad/12/c/148236593/large/2.jpg?dt=803fe913c276202fc39751444debda98

[27] Autodesk.com - Aspectos generales de AutodeskFusion 360 (Última vez visitado: 18 de junio de 2017): https://www.autodesk.com/products/fusion-360/overview

[28] Autodesk.com - Soporte Técnico Autodesk – Formatos de exportación soportados por Autodesk Fusion 360 (Última vez visitado: 18 de junio de 2017): https://knowledge.autodesk.com/es/support/fusion-360/learn-explore/caas/sfdcarticles/sfdcarticles/ESP/Export-format-options-for-Fusion-360.html

[29] Ubuntu.com – Historia de Ubuntu. (Última vez visitado: 24 de junio de 2017): https://www.ubuntu.com/about/about-ubuntu

[30] Ros.org - Características de ROS (Última vez visitado: 24 de junio de 2017): http://wiki.ros.org/ROS/Concepts

[31] ROS Morgan Quigley, Brian Gerkey, Ken Conley, Josh Faust, Tully Foote, Jeremy Leibs, Eric Berger, Rob Wheeler, Andrew Ng. «ROS: an open-source Robot Operating System».

[32] Prats, M.; Perez, J.; Fernandez, J.J.; Sanz, P.J., "An open source tool for simulation and supervision of underwater intervention missions", 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 2577-2582, 7-12 Oct. 2012

[33] IRS.uji.es – Acerca de UWSim. (Última vez visitado: 24 de junio de 2017): www.irs.uji.es/uwsim/about

[34] IRS.uji.es – Características principales de UWSim. (Última vez visitado: 24 de junio de 2017): www.irs.uji.es/uwsim/uwsim-features

[35] IRS.uji.es – Características del sensor multihaz. (Última vez visitado: 24 de junio de 2017): www.irs.uji.es/uwsim/news/multibeam-sensor

[36] Despacho de abogados internacional de derecho empresarial Garrigues – Regulación de los drones marinos (Última vez visitado: 23 de julio de 2017): http://www.garrigues.com/es_ES/noticia/la-regulacion-de-los-drones-marinos

[37] Bluerobotics.com – Bluerov2 imágenes (Última vez visitado: 3 de agosto de 2017): http://www.bluerobotics.com/store/rov/bluerov2/

[38] Bluerobotics.com - Datasheet del BlueROV2 (Última vez visitado: 3 de agosto de 2017): http://bluerobotics.com/downloads/bluerov2.pdf

[39] Grabcad.com – Descarga del modelo 3D del BlueROV2. (Última vez visitado: 28 de junio de 2017): https://grabcad.com/library/bluerov2-1

[40] Ros.org - Descripción y primeros pasos con un archivo URDF (Última vez visitado: 8 de julio de 2017): http://wiki.ros.org/urdf/Tutorials/Create%20your%20own%20urdf%20file

[41] IRS.uji.es – Configurar una escena para UWSim. (Última vez visitado: 22 de julio de 2017): http://www.irs.uji.es/uwsim/wiki/index.php?title=Configuring_and_creating_scenes

[42] Ubuntu-guía.com - Guía de instalación de Ubuntu 14.04 (Última vez visitado: 5 de mayo de 2017): http://www.ubuntu-guia.com/2012/04/como-instalar-ubuntu.html

[43] Ubuntu.com – Repositorio de descarga de la versión 14.04 Indigo de Ubuntu. (Última vez visitado: 5 de mayo de 2017): http://releases.ubuntu.com/14.04/

[44] Wiki.ros.org - Guía de instalación de ROS Indigo (Última vez visitado: 6 de mayo de 2017): wiki.ros.org/indigo/installation/Ubuntu

[45] Ubuntu.com - Repositorios de Ubuntu 14.04.5 LTS (Última vez visitado: 5 de mayo de 2017): help.ubuntu.com/community/repositories/Ubuntu

[46] IRS.uji.es – Instalación de UWSim en Linux. (Última vez visitado: 7 de mayo de 2017): www.irs.uji.es/uwsim/wiki/index.php?title=Installing_UWSim

Índice de figuras

Índice de tablas